DISCIPLINA

FRONT-END COM HTML, CSS, JAVASCRIPT E ANGULAR

SUMÁRIO

O que é HTML? 11
O que você pode fazer com HTML? 11
HTML Primeiros passos 12
Começando 12
Criando Seu Primeiro Documento HTML 12
Etapa 1: Criação do arquivo HTML 12
Etapa 2: digite algum código HTML 12
Etapa 3: Salvar o arquivo 12
Explicação do código 12
Tags e elementos HTML 13
Elementos HTML 14
Sintaxe do elemento HTML 14
Tags HTML vs Elementos 14
Elementos HTML vazios 14
Elementos HTML de aninhamento 15
Escrevendo comentários em HTML 15
Tipos de elementos HTML 16
Atributos HTML 17
O que são atributos 17
Atributos de uso geral 18
O atributo id 18
O atributo de classe 18
O atributo title (*título*) 18
O atributo de style (*estilo*) 19
Cabeçalhos HTML 20
Organização de conteúdo com títulos 20
Importância dos títulos 21
Parágrafos HTML 21
Criação de parágrafos 21
Criando Quebras de Linha 22
Criação de regras horizontais 22

Gerenciando Espaços Brancos 22
Definindo Texto Pré-formatado 23
Links HTML 23
Criação de links em HTML 23
Sintaxe do link HTML 23
Definindo os alvos para links 24
Criação de âncoras de marcador 24
Criação de links de download 25
Formatação de Texto HTML 25
Formatando Texto com HTML 25
Diferença entre e <b> 26
Diferença entre e <i> 26
Formatando citações 26
Mostrando abreviações 26
Marcando endereços de contato 27
Estilos HTML 27
Estilo de elementos HTML 27
Adicionando estilos a elementos HTML 27
Estilos embutidos (inline) 27
Folhas de estilo incorporadas (embedded) 28
Folhas de estilo externas 28
Vinculando folhas de estilo externas 28
Importando folhas de estilo externas 29
Imagens HTML 30
Inserindo imagens em páginas da web 30
Definindo a largura e altura de uma imagem 30
Usando o elemento de imagem HTML5 31
Trabalhando com Mapas de Imagens 31
Tabelas HTML 32
Criação de tabelas em HTML 32
Abrangendo várias linhas e colunas 33
Adicionar legendas às tabelas 33
Definindo um cabeçalho, corpo e rodapé de tabela 33
Listas HTML 34
Trabalho com listas de HTML 34
Listas HTML não ordenadas 34

Listas ordenadas em HTML 35
Listas de descrição de HTML 35
Formulários HTML 36
O que é um formulário HTML? 36
Elemento de entrada 36
Campos de Texto 36
Campo de senha 36
Botões do rádio 36
Caixas de seleção 37
Caixa de seleção de arquivo 37
Textarea 37
Caixas de Seleção 37
Botões de envio e redefinição 38
Controles de formulário de agrupamento 38
Atributos de formulário usados com frequência 38
HTML Meta 39
Definindo Metadados 39
Declaração de codificação de caracteres em HTML 39
Definindo o autor de um documento 39
Palavras-chave e descrição para motores de busca 39
Configurando a janela de visualização (Viewport) para dispositivos móveis 39
Introdução CSS 41
O que você pode fazer com CSS 41
Vantagens de usar CSS 41
Seletores CSS 42
O que é Seletor? 42
Seletor Universal 42
Seletores de tipo de elemento 43
Seletores de Id 43
Seletores de classe 43
Seletores de descendentes 43
Seletores filhos 44
Seletores de irmãos adjacentes 44
Seletores gerais de irmãos 44
Seletores de agrupamento 45
CSS Color 45

Configurando a propriedade da cor 45
Definindo Valores de Cor 45
Palavras-chave de cor 46
Valores de cor HEX 46
Valores de cor RGB 46
Aplicação da propriedade da cor nas bordas e contornos 46
Fundo CSS – Background 47
Configurando propriedades de fundo 47
Cor de fundo – (color) 47
Imagem de fundo – (image) 47
Repetição de imagem de fundo – (repeat) 47
Posição de fundo – (position) 48
Anexo de Fundo – (attachment) 49
A propriedade curta de definição de fundo – (shorthand) 49
Fontes CSS 50
Fontes de estilo com CSS 50
Família de fontes – (family) 50
Diferença entre fontes Serif e Sans-serif 51
Estilo de fonte – (style) 51
Tamanho da fonte – (size) 51
Configurando o tamanho da fonte com pixels 51
Configurando o tamanho da fonte com EM 52
Usando a combinação de porcentagem e EM 52
Configurando o tamanho da fonte com Root EM 52
Configurando o tamanho da fonte com palavras-chave 53
Configurando o tamanho da fonte com unidades de viewport 53
Espessura da fonte – (weight) 53
Variante de fonte – (variant) 54
Texto CSS 54
Formatando Texto com CSS 54
Cor do texto 54
Alinhamento de Texto 54
Decoração de Texto 55
Removendo o sublinhado padrão dos links 55
Transformação de Texto 55
Recuo de Texto 55

Espaçamento entre Letras 56
Espaçamento entre Palavras 56
Altura da linha 56
Links CSS 56
Links de estilo com CSS 56
Modificando Estilos de Link Padrão 57
Configurando a cor personalizada dos links 57
Removendo o sublinhado padrão dos links 58
Fazendo os links de texto parecerem botões 58
Listas CSS 59
Tipos de listas HTML 59
Listas de estilo com CSS 59
Alterar o tipo de marcador das listas 59
Alterar a posição dos marcadores de lista 59
Usando imagens como marcadores de lista 60
Configurando todas as propriedades da lista de uma vez 60
Criação de menus de navegação usando listas 61
Tabelas CSS 61
Tabelas de estilo com CSS 61
Adicionando Bordas a Tabelas 61
Recolhendo bordas da tabela 62
Ajustando o Espaço dentro das Tabelas 62
Definir a largura e altura da mesa 63
Controlando o Layout da Tabela 63
Alinhando o texto dentro das células da tabela 63
Alinhamento horizontal do conteúdo da célula 64
Alinhamento vertical do conteúdo da célula 64
Controlando a posição da legenda da tabela 64
Manipulação de células vazias 64
Criação de tabelas listradas (zebradas) 65
Tornando uma Tabela Responsiva 65
CSS Box Model 65
O que é Box Model? 65
Largura e altura dos elementos 66
Dimensão CSS 67
Definindo as dimensões do elemento 67

Definir a largura e altura 67
Definir largura e altura máximas 67
Definir largura e altura mínimas 68
Definir um intervalo de largura e altura 68
Padding CSS 68
Propriedades de preenchimento CSS 68
Definir Paddings para Lados Individuais 69
A propriedade Padding Shorthand 69
Efeito de preenchimento e borda no layout 70
Borda CSS 70
Propriedades de borda CSS 70
Compreendendo os diferentes estilos de borda 70
Configurando a largura da borda 71
Especificando a cor da borda 71
A propriedade da abreviação de borda 71
Margem CSS 71
Propriedades de margem CSS 71
Definindo Margens para Lados Individuais 72
A propriedade Margin Shorthand 72
Centralização horizontal com margens automáticas 73
Tutorial de JavaScript 74
O que você pode fazer com JavaScript 74
Introdução ao JavaScript 74
Adicionando JavaScript às suas páginas da web 74
Incorporando o código JavaScript 75
Chamando um arquivo JavaScript externo 75
Colocando o código JavaScript embutido 76
Posicionamento do script dentro do documento HTML 76
Diferença entre scripts do lado do cliente e do lado do servidor 76
Sintaxe JavaScript 77
Compreendendo a sintaxe do JavaScript 77
Sensibilidade a maiúsculas e minúsculas em JavaScript 77
Comentários de JavaScript 77
Variáveis JavaScript 78
O que é variável? 78
Declarando múltiplas variáveis de uma só vez 78

As palavras-chave let e const (ES6) 78
Convenções de nomenclatura para variáveis JavaScript 79
Saída de geração de JavaScript 79
Gerando saída em JavaScript 79
Gravando saída no console do navegador 79
Exibindo a saída em caixas de diálogo de alerta 79
Gravando a saída na janela do navegador 79
Inserindo saída dentro de um elemento HTML 80
Tipos de dados JavaScript 80
O tipo de dados String 80
O tipo de dados do número 81
O tipo de dados booleano 81
O tipo de dado indefinido 81
O tipo de dado nulo 81
O tipo de dados do objeto 81
O tipo de dados Array 82
O tipo de dados Função 82
O tipo de operador 82
Operadores de JavaScript 83
Operadores aritméticos JavaScript 84
Operadores de atribuição de JavaScript 84
Operadores de string JavaScript 84
Operadores de incremento e decremento de JavaScript 84
Operadores lógicos de JavaScript 85
Operadores de comparação de JavaScript 85
Eventos JavaScript 85
Compreendendo eventos e manipuladores de eventos 85
Eventos de mouse 86
O evento Click (onclick) 86
O evento Contextmenu (oncontextmenu) 86
O evento Mouseover (onmouseover) 86
O evento Mouseout (onmouseout) 86
Eventos de teclado 87
O evento Keydown (onkeydown) 87
O evento Keyup (onkeyup) 87
O evento Keypress (onkeypress) 87

Eventos de formulário 87
O Evento Focus (onfocus) 87
O evento de desfoque (onblur) 87
O Evento de Mudança (onchange) 87
O evento de envio (onsubmit) 87
Eventos de documento / janela 88
O evento de carregamento (onload) 88
O evento de descarregamento (onunload) 88
O evento de redimensionamento (onresize) 88
JavaScript Strings 88
Sequências de escape de JavaScript 88
Executando Operações em Strings 88
Obtendo o comprimento de uma string 89
Encontrando uma string dentro de outra string 89
Procurando um padrão dentro de uma string 89
Extraindo uma substring de uma string 89
Extração de um número fixo de caracteres de uma string 90
Substituindo o conteúdo de uma string 90
Converter uma string em maiúsculas ou minúsculas 90
Acessando caracteres individuais de uma string 90
Dividindo uma String em uma Matriz 90
Números de JavaScript 90
Trabalhando com Números 91
Evitando problemas de precisão 91
Execução de operações em números 91
Analisando inteiros de strings 91
Convertendo Números em Strings 91
Formatando Números para Decimais Fixos 92
Formatando Números com Precisão 92
Declarações JavaScript If ... Else 92
A declaração “if” 92
A declaração – if / else 92
A declaração if / elseif / else 93
O operador ternário 93
Switch...Case 93
Usando o Switch /Case Statement 93

Vários casos compartilhando a mesma ação 93
O que é o Angular? 94
Aplicações angulares: o essencial 94
Componentes 94
Modelos 95
Property Bindings 96
Event Listeners 96
Diretivas 98
Injeção de dependência 98
Angular CLI 100
Bibliotecas primárias 100

O que é HTML?

HTML significa HyperText Markup Language. Ela é a linguagem de construção básica da World Wide Web.

HyperText ou Hipertexto é o texto exibido em um computador ou outro dispositivo eletrônico com referências a outro texto que o usuário pode acessar imediatamente, geralmente com um clique do mouse ou pressionamento de tecla.

Além do texto, o hipertexto pode conter tabelas, listas, formulários, imagens e outros elementos de apresentação. É um formato fácil de usar e flexível para compartilhar informações pela Internet.

As linguagens de marcação usam conjuntos de tags de marcação para caracterizar os elementos de texto em um documento, o que fornece instruções aos navegadores da web sobre como o documento deve aparecer.

HTML foi originalmente desenvolvido por Tim Berners-Lee em 1990. Ele também é conhecido como o pai da web. Em 1996, o World Wide Web Consortium (W3C) tornou-se a autoridade para manter as especificações HTML. O HTML também se tornou um padrão internacional (ISO) em 2000.

HTML5 é a versão mais recente do HTML. O HTML5 fornece uma abordagem mais rápida e robusta para o desenvolvimento da web.

O que você pode fazer com HTML?

Existem muito mais coisas que você pode fazer com HTML.

- Você pode publicar documentos online com texto, imagens, listas, tabelas, etc.
- Você pode acessar recursos da web, como imagens, vídeos ou outros documentos HTML por meio de hiperlinks.
- Você pode criar formulários para coletar entradas do usuário, como nome, endereço de e-mail, comentários, etc.
- Você pode incluir imagens, vídeos, clipes de som, filmes em flash, aplicativos e outros documentos HTML diretamente dentro de um documento HTML.
- Você pode criar uma versão offline do seu site que funcione sem internet.
- Você pode armazenar dados no navegador da web do usuário e acessá-los posteriormente.
- Você pode encontrar a localização atual do visitante do seu site.

A lista não termina aqui, há muitas outras coisas interessantes que você pode fazer com HTML.

> **Nota:** HTML conforme descrito anteriormente é uma linguagem de marcação, não uma linguagem de programação, como Java, Ruby, PHP, etc. Você precisa de um navegador da web para visualizar as páginas HTML. Os navegadores da web não exibem as tags HTML, mas usam as tags para interpretar o conteúdo das páginas da web.

HTML Primeiros passos

Começando

Neste tutorial, você aprenderá como é fácil criar um documento HTML ou uma página da web. Para começar a codificar HTML, você só precisa de duas coisas: um editor de texto simples e um navegador da web.

Bem, vamos começar a criar sua primeira página HTML.

Criando Seu Primeiro Documento HTML

Vamos percorrer as etapas a seguir. No final deste tutorial, você terá feito um arquivo HTML que exibe a mensagem "Olá, mundo" em seu navegador da web.

Etapa 1: Criação do arquivo HTML

Abra o editor de texto simples do seu computador e crie um novo arquivo.

Etapa 2: digite algum código HTML

Comece com uma janela vazia e digite o seguinte código:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Um simples documento HTML</title>
</head>
<body>
    <p>Olá mundo!!<p>
</body>
</html>
```

Figura 1 - HTML: Meu primeiro arquivo

Etapa 3: Salvar o arquivo

Agora salve o arquivo em sua área de trabalho (ou em uma pasta qualquer de sua escolha) como "index.html".

Para abrir o arquivo em um navegador., navegue até o seu arquivo e clique duas vezes nele. Ele será aberto em seu navegador da Web padrão, caso contrário, abra seu navegador e arraste o arquivo para ele.

Explicação do código

Você pode pensar do que se tratava esse código. Bem, vamos descobrir.

- A primeira linha `<!DOCTYPE html>` é a declaração do tipo de documento. Ele instrui o navegador da web que este documento é um documento HTML5. Não há distinção entre maiúsculas e minúsculas.

- O elemento `<head>` é um contêiner para as tags que fornecem informações sobre o documento, já a tag `<title>` define o título do mesmo.

- O elemento `<body>` contém o conteúdo real do documento (parágrafos, links, imagens, tabelas e assim por diante) que é renderizado no navegador da web e exibido para o usuário.

Você aprenderá sobre os diferentes elementos HTML em detalhes nos próximos capítulos. Por enquanto, concentre-se apenas na estrutura básica do documento HTML.

> **Dica:** Os elementos `<html>`, `<head>` e `<body>` compõem o esqueleto básico de uma página web. O conteúdo dentro do `<head>` e `</head>` é invisível para os usuários, com uma exceção: o texto entre as tags `<title>` e `</title>`que aparece como título em uma guia do navegador.

Tags e elementos HTML

O HTML é escrito na forma de elementos que consistem em tags de marcação. Essas tags de marcação são a característica fundamental do HTML. Cada tag de marcação é composto por uma palavra-chave, cercado por colchetes, tais como `<html>`, `<head>`, `<body>`, `<title>`, `<p>`, e assim por diante.

As tags HTML normalmente vêm em pares como `<html>` e `</html>`. A primeira tag de um par é frequentemente chamada de tag de abertura ou tag inicial, e a segunda tag é chamada de tag de fechamento ou tag final.

Uma tag de abertura e uma tag de fechamento são idênticas, exceto por uma barra (`/`) após o colchete angular de abertura da tag de fechamento, para informar ao navegador que o comando foi concluído.

Entre as tags de início e fim, você pode colocar o conteúdo apropriado. Por exemplo, um parágrafo, que é representado pelo `<p>` elemento, seria escrito como:

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <title>Um simples documento HTML</title>
5  </head>
6  <body>
7      <p>Isto é um parágrafo.</p>
8      <!-- Parágrafo com elemento aninhado -->
9      <p>
10         Este é <b>outro</b> parágrafo.
11     </p>
12 </body>
13 </html>
```

Figura 2 - HTML: Exemplo de um parágrafo.

Elementos HTML

Sintaxe do elemento HTML

Um elemento HTML é um componente individual de um documento HTML. Representa semântica ou significado. Por exemplo, o elemento `title` representa o título do documento.

A maioria dos elementos HTML são escritos com uma *tag de início* ou tag de abertura e uma *tag de finalização* ou tag de fechamento, com conteúdo intermediário. Os elementos também podem conter atributos que definem suas propriedades adicionais.

Por exemplo, um parágrafo, que é representado pelo elemento `<p>`, seria escrito como:

Figura 3 - Exemplo da tag de parágrafo com atributos

Tags HTML vs Elementos

Tecnicamente, um elemento HTML é a coleção da tag de início, seus atributos, uma tag de finalização e tudo mais. Por outro lado, uma tag HTML (de abertura ou fechamento) é usada para marcar o início ou o fim de um elemento, como você pode ver na ilustração acima.

No entanto, no uso comum, os termos elemento HTML e tag HTML são intercambiáveis, ou seja, uma tag é um elemento. Para simplificar, os termos "tag" e "elemento" têm o mesmo significado - já que definirão algo em sua página da web.

No HTML, as tags não diferenciam maiúsculas de minúsculas (mas a maioria dos valores de atributo diferencia maiúsculas de minúsculas). Ou seja, tanto a tag `<P>`, como a tag `<p>` irão definir a mesma coisa, ou seja, um parágrafo.

Elementos HTML vazios

Elementos vazios (também chamados de elementos de fechamento automático ou nulos) não são tags de contêiner - isso significa que você não pode escrever.
`<hr>conteúdo</hr> - <br>outro conteúdo</br>`

Um exemplo típico de um elemento vazio é o `<br>`, que representa uma quebra de linha. Alguns outros elementos vazios são comuns `<img>`, `<input>`, `<link>`, `<meta>`, `<hr>`, etc.

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Entendendo os elementos HTML</title>
</head>
<body>
  <p>Este parágrafo contém <br> uma quebra de linha.</p>
  <img src="images/sky.jpg" alt="Céu Azul">
  <input type="text" name="usuario">
</body>
</html>
```

Figura 4 - HTML: Elementos com e sem fechamento de tag

Elementos HTML de aninhamento

A maioria dos elementos HTML pode conter qualquer número de outros elementos (exceto elementos vazios), que são, por sua vez, compostos de tags, atributos e conteúdo ou outros elementos.

O exemplo a seguir mostra alguns elementos aninhados dentro do elemento `<p>`.

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Entendendo os elementos HTML</title>
</head>
<body>
  <p>Isto é um texto em <b>negrito</b>.</p>
  <p>Isto é um texto com texto em <i>itálico</i>.</p>
  <p>Isto é um texto com texto <mark>"marcado"</mark>.</p>
</body>
</html>
```

Figura 5 - HTML: Elementos aninhados

As tags HTML devem ser aninhadas na ordem correta. Devem ser fechados na ordem inversa de como são definidos, ou seja, o último tag aberto deve ser fechado primeiro.

```
<p>Isto é um texto em <b>negrito</b>.</p>
<p>Isto é um texto com texto em <i>itálico</i>.</p>
<p>Isto é um texto com texto <mark>"marcado"</mark>.</p>
```

Figura 6 - HTML - Respeitando a ordem de fechamento das tags

Escrevendo comentários em HTML

Os comentários são geralmente adicionados com o objetivo de tornar o código-fonte mais fácil de entender. Isso pode ajudar outro desenvolvedor (ou você no futuro, ao editar o código-fonte) a entender o que você estava tentando fazer com o HTML. Os comentários não são exibidos no navegador.

Um comentário HTML começa `<!--` e termina com `-->`, conforme mostrado no exemplo abaixo:

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Comentários em HTML</title>
</head>
<body>
  <!-- Isto é um comentário em HTML -->
  <!-- Os comentários em HTML podem usar uma
       ou várias linhas, desde que organizadas corretamente -->
  <p>Isto é um pedaço normal de texto.</p>
</body>
</html>
```

Figura 7 - HTML: Escrevendo comentários em HTML

Você também pode comentar parte do seu código HTML para fins de depuração, conforme mostrado aqui:

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Usando comentário para ocultar elementos HTML</title>
</head>
<body>
  <!-- Ocultando uma tag de imagem para teste
    <img src="images/smiley.png" alt="Smiley">
  -->
</body>
</html>
```

Figura 8 - HTML: Ocultando parte do código utilizando comentários

Tipos de elementos HTML

Os elementos podem ser colocados em dois grupos distintos: *elementos de nível de bloco* e elementos de *nível embutido*.

Os primeiros constituem a estrutura do documento, enquanto os segundos revestem o conteúdo de um bloco.

Além disso, um elemento de bloco ocupa 100% da largura disponível e é renderizado com uma quebra de linha antes e depois, já um elemento embutido ocupará apenas o espaço de que necessita.

Os elementos de bloco mais utilizados são `<div>`, `<p>`, `<h1>` até o `<h6>`, `<form>`, `<ol>`, `<ul>`, `<li>`, e assim por diante.

Os elementos de nível embutido mais comumente utilizados são `<img>`, `<a>`, `<span>`, `<strong>`, `<b>`, `<em>`, `<i>`, `<code>`, `<input>`, `<button>`, etc.

Atributos HTML

O que são atributos

Os atributos definem características ou propriedades adicionais do elemento, como largura e altura de uma imagem.

Os atributos são sempre especificados na tag de início e geralmente consistem em pares de nome / valor como `name="value"`. Os valores dos atributos devem sempre ser colocados entre aspas.

Além disso, alguns atributos são necessários para certos elementos. Por exemplo, uma tag `<img>` deve conter os atributos `src` e `alt`.

Vamos dar uma olhada em alguns exemplos de usos de atributos:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Elementos com atributos obrigatórios</title>
</head>
<body>
  <img src="images/smiley.png" width="30" height="30" alt="Smiley">
  <a href="https://www.google.com/" title="Buscador Google">Google</a>
  <abbr title="Hyper Text Markup Language">HTML</abbr>
  <input type="text" value="João">
</body>
</html>
```

Figura 9 - HTML: Elementos com atributos obrigatórios

No exemplo acima, o atributo `src` da tag `<img>` é um atributo que define o caminho da imagem, e seu valor (endereço da imagem) é fornecido dentro das aspas. Da mesma forma, o atributo `href` dentro da tag `<a>` fornece o link do endereço para onde o visitante será direcionado ao clicar na âncora que estará disponível no valor que fica entre a tag de abertura e fechamento (“Google”, neste exemplo).

Existem vários atributos em HTML5 que não consistem em pares nome/valor, mas consistem apenas em nome. Esses atributos são chamados de atributos booleanos. Exemplos de alguns atributos booleanos utilizados são `checked`, `disabled`, `readonly`, `required`, etc.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Elementos com atributos obrigatórios</title>
</head>
<body>
  <input type="email" required>
  <input type="submit" value="Enviar" disabled>
  <input type="checkbox" checked>
  <input type="text" value="Texto somente leitura" readonly>
</body>
</html>
```

Figura 10 - HTML: Atributos somente chave (sem valor)

Atributos de uso geral

Existem alguns atributos, como `id`, `title`, `class`, `style`, etc, que você pode usar na maioria dos elementos HTML.

O atributo id

O atributo `id` é usado para fornecer um nome ou identificador **exclusivo** a um elemento em um documento. Isso torna mais fácil selecionar o elemento usando CSS ou JavaScript.

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Atributos únicos definidos pelo "id"</title>
</head>
<body>
  <input type="text" id="nome">
  <div id="container">Conteúdo</div>
  <p id="infoText">Isto é um parágrafo.</p>
</body>
</html>
```

Figura 11 - HTML - Parametrização de identificação única de tag com id

O atributo de classe

Como o atributo `id`, o atributo `class` também é usado para identificar elementos. Mas `id`, ao contrário do atributo `class` não precisa ser exclusivo no documento. Isso significa que você pode aplicar a mesma classe a vários elementos em um documento, conforme mostrado no exemplo a seguir:

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Atributos únicos definidos pelo "id"</title>
</head>
<body>
  <input type="text" id="nome" class="highlight">
  <div id="container" class="box highlight">Conteúdo</div>
  <p id="infoText" class="highlight">Isto é um parágrafo.</p>
</body>
</html>
```

Figura 12 - HTML: Atribuição de identificação de classe a várias tags diferentes usando o atributo class

O atributo title (*título*)

O atributo `title` é usado para fornecer um texto consultivo sobre um elemento ou seu conteúdo. O valor do atributo title é exibido como uma "dica" de ferramenta pelos navegadores da web quando o usuário posiciona o cursor do mouse sobre o elemento.

Importante salientar e se atentar que o Google aumentou recentemente o comprimento dos títulos em suas páginas de resultados do mecanismo de pesquisa (SERPs) de 50-60 caracteres para aproximadamente 70 caracteres, desde que o comprimento do conteúdo da tag de título não exceda 600 pixels.

O atributo de style (*estilo*)

O atributo `style` permite que você especifique regras de estilo CSS, como cor, fonte, borda, etc., diretamente no elemento. Vamos dar uma olhada em um exemplo para ver como funciona:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Atributos únicos definidos pelo "id"</title>
</head>
<body>
  <input type="text" id="nome" style="border: 1px solid red;">
  <div id="container" style="width: 300px;">Conteúdo</div>
  <p id="infoText" style="color: blue;">Isto é um parágrafo.</p>
</body>
</html>
```

Figura 13 - HTML: Aplicando estilização diretamente numa tag através de style

Os atributos que discutimos acima também são chamados de atributos globais. Além dos atributos específicos do elemento, o HTML5 define alguns atributos que são comuns a todos os elementos. Estes atributos podem ser especificados em todos os elementos, com algumas exceções em que não são relevantes, tais como, elementos encontrados no interior da seção <head> do documento, por exemplo <base>, <script>, <title>, etc.

Tabela 1 - HTML: Atributos globais do HTML5

Atributo	Valor	Descrição
accesskey	*tecla de atalho*	Especifica um atalho de teclado para ativar ou focar o elemento.
class	*nome da classe*	Atribui um nome de classe ou lista separada por espaço de nomes de classe a um elemento.
contenteditable	`true` `false`	Indica se o conteúdo de um elemento pode ser editado pelo usuário ou não.
contextmenu	*menu-id*	Especifica um menu de contexto para um elemento. Um menu de contexto é um menu que aparece quando o usuário clica com o botão direito do mouse no elemento.
data-*	*dados*	Especificado em qualquer elemento HTML, para armazenar dados personalizados específicos da página.
dir	`ltr` `rtl`	Especifica a direção básica da direcionalidade do texto do elemento.
draggable	`true` `false`	Especifica se um elemento pode ser arrastado ou não.

dropzone	`copy`
	`move`
	`link`
		Especifica se os dados arrastados são copiados, movidos ou vinculados, quando soltos.

hidden	`hidden`	Indica que o elemento ainda não é ou não é mais relevante.
id	*nome*	Especifica um identificador único (ID) para um elemento que deve ser único em todo o documento.

lang	*código de linguagem*	Especifica o idioma principal para o conteúdo de texto do elemento.
spellcheck	`true` `false`	Especifica se o elemento pode ser verificado em busca de erros ortográficos ou não.
style	*estilo*	Especifica informações de estilo embutido para um elemento.
tabindex	*número*	Especifica a ordem de tabulação de um elemento.
title	*texto*	Fornece informações consultivas relacionadas ao elemento. Seria apropriado para uma dica de ferramenta.
translate	yes no	Especifica se o conteúdo do texto de um elemento deve ser traduzido ou não.
xml:lang	*código de linguagem*	Especifica o idioma principal para o conteúdo de texto do elemento, em documentos XHTML.

Cabeçalhos HTML

Organização de conteúdo com títulos

Os títulos ajudam a definir a hierarquia e a estrutura do conteúdo da página da web.

HTML oferece seis níveis de tags de título que vão do `<h1>` até o `<h6>`;
Quanto mais alto o número do nível do cabeçalho, maior sua importância - portanto, a tag `<h1>` define o cabeçalho mais importante, enquanto a tag `<h6>` define o cabeçalho menos importante no documento.

Por padrão, os navegadores exibem cabeçalhos em fontes maiores e mais em negrito do que o texto normal. Logo, os títulos `<h1>` são exibidos na fonte maior, enquanto os títulos `<h6>` são exibidos na fonte menor.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Hierarquia e comportamento dos Cabeçalhos HTML</title>
</head>
<body>
  <h1>Heading level 1</h1>
  <h2>Heading level 2</h2>
  <h3>Heading level 3</h3>
  <h4>Heading level 4</h4>
  <h5>Heading level 5</h5>
  <h6>Heading level 6</h6>
</body>
</html>
```

Figura 14 - HTML: Comportamento da tag de cabeçalho (<h1> até <h6>)

- A saída do exemplo acima será semelhante a esta:

Figura 15 - HTML: Resultado da aplicação das tags de cabeçalho

Importante salientar que cada vez que você coloca uma tag de título em uma página da web, as folhas de estilo integradas do navegador da web criam automaticamente algum espaço vazio (denominado margem) antes e depois de cada título. Você pode usar a propriedade `margin` do CSS para substituir a folha de estilo padrão do navegador.

Importância dos títulos

- Os títulos HTML fornecem informações valiosas, destacando tópicos importantes e a estrutura do documento, portanto, otimize-os cuidadosamente para melhorar o envolvimento do usuário.
- Não use títulos para fazer seu texto parecer GRANDE ou em negrito. Use-os apenas para destacar o título do seu documento e para mostrar a estrutura do documento, já que os motores de busca, como o Google, usam estes mesmos cabeçalhos para indexar a estrutura e o conteúdo das páginas da web, portanto, use-os com muita sabedoria em sua página da web.
- Use os cabeçalhos `<h1>` como cabeçalhos principais de sua página da web, seguidos pelos cabeçalhos `<h2>`, depois os cabeçalhos `<h3>` menos importantes e assim por diante.

Parágrafos HTML

Criação de parágrafos

Como visto nos exemplos anteriores, o elemento de parágrafo é usado para publicar texto nas páginas da web.

Os parágrafos são definidos com a tag `<p>`. A tag de parágrafo é muito básica e normalmente é a primeira tag que você precisará para publicar seus textos nas páginas da web.

As folhas de estilo integradas dos navegadores criam automaticamente um espaço acima e abaixo do conteúdo de um parágrafo (denominado margem), mas você pode substituí-lo usando CSS.

Criando Quebras de Linha

A tag `<br>` é usada para inserir uma quebra de linha na página da web.

Como o `<br>` é um elemento vazio, **não há necessidade de fecha-lo** utilizando `</br>`.

IMPORTANTE! Não use o parágrafo vazio, ou seja, " " para adicionar espaço extra em suas páginas da web. O navegador pode ignorar os parágrafos vazios, pois é uma tag lógica! Ao invez disso, use a propriedade CSS *margin* para ajustar o espaço ao redor dos elementos.

Criação de regras horizontais

Você pode usar a tag `<hr>` para criar linhas ou linhas horizontais para separar visualmente as seções de conteúdo em uma página da web. Como `<br>`, a tag `<hr>` também é um elemento vazio.

Gerenciando Espaços Brancos

Normalmente, o navegador exibirá os vários espaços criados dentro do código HTML pressionando a tecla da *barra de espaço* ou a *tecla tab* no teclado como um único espaço. Várias quebras de linha criadas dentro do código HTML pressionando a tecla "Enter" também são exibidas como um único espaço.

Para contornar esta situação e inserir espaços em branco consecutivos entre palavras ou elementos, utilize *** ***, e se quiser quebrar linhas, lembre-se de utilizar a tag `<br>`

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <title>Espaço em branco num parágrafo</title>
5  </head>
6  <body>
7    <p>
8      Este parágrafo tem multiplos     espaços entre a frase.
     </p>
9    <p>
10     Este parágrafo tem multiplas<br><br>quebras<br><br>de<br>linhas.
11   </p>
12 </body>
13 </html>
```

Figura 16 - HTML: Inserindo espaços em brancos válidos para o navegador

Definindo Texto Pré-formatado

Às vezes porém, usar ` `, `<br>` para o gerenciamento de espaços não é muito conveniente. Alternativamente, você pode usar a tag `<pre>` para exibir espaços, tabulações, quebras de linha, etc. exatamente como escrito no arquivo HTML. É muito útil na apresentação de texto onde os espaços e as quebras de linha são importantes, como poemas ou códigos.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Exemplo de exibição de texto pré-formatado</title>
</head>
<body>
  <pre>
      Brilha, brilha, estrelinha,
      Quero ver você brilhar,
      Lá no alto, lá no céu,
      Num desenho de cordel...
  </pre>
</body>
</html>
```

Figura 17 - HTML: Texto pre-formatado usando <pre>

Links HTML

Criação de links em HTML

Um link ou hiperlink é uma conexão de um recurso da web para outro. Os links permitem que os usuários passem facilmente de uma página para outra, em qualquer servidor em qualquer lugar do mundo.

Um link possui duas extremidades, chamadas âncoras. O link começa na âncora de origem e aponta para a âncora de destino, que pode ser qualquer recurso da web, por exemplo, uma imagem, um áudio ou videoclipe, um arquivo PDF, um documento HTML ou um elemento dentro do próprio documento e assim sobre.

Por padrão, os links aparecerão da seguinte forma na maioria dos navegadores:

- Um link não visitado está sublinhado e em azul.
- Um link visitado está sublinhado e roxo.
- Um link ativo está sublinhado e vermelho.

No entanto, você pode sobrescrever isso usando CSS.

Sintaxe do link HTML

Os links são especificados em HTML usando a tag `<a>`.

Um link ou hiperlink pode ser uma palavra, grupo de palavras ou imagem.

```
<a href="url"> Texto do link </a>
```

Qualquer coisa entre a tag `<a>` de abertura e a tag `</a>` de fechamento se torna a parte do link que o usuário vê e clica em um navegador.

```
6  <body>
7    <a href="https://www.google.com/">Google</a>
8    <a href="https://uniciv.com.br/">UNICIV - Pós Graduação EAD em TI</a>
9    <a href="images/cats.jpg">
10       <img src="cats-thumb.jpg" alt="cats">
11   </a>
12 </body>
```

Figura 18 - HTML: Exemplos de links

O atributo `href` especifica o destino do link. Seu valor pode ser um URL absoluto ou relativo.

Um URL absoluto é o URL que inclui todas as partes do formato de URL, tal como o protocolo, nome de host, e o caminho do documento, por exemplo:

- `https://www.google.com/`
- `https://www.dominio.com.br/form.php`

Um URL relativos são caminhos da página-relativa, ou seja, páginas do próprio contexto do website, por exemplo:

- `contato.html`
- `images/cats.jpg`

Um URL relativo nunca inclui o prefixo `http://` ou `https://`

Definindo os alvos para links

O atributo `target` informa ao navegador onde abrir o documento vinculado. Existem quatro destinos definidos, e cada nome de destino começa com um caractere sublinhado (`_`):

- `_blank` - Abre o documento vinculado em uma nova janela ou guia.
- `_parent` - Abre o documento vinculado na janela pai.
- `_self`- Abre o documento vinculado na mesma janela ou guia do documento de origem. Este é o padrão, portanto, não é necessário especificar explicitamente esse valor.
- `_top` - Abre o documento vinculado na janela inteira do navegador.

```
6  <body>
7    <a href="/sobre-nos.php" target="_top">Sobre Nós</a>
8    <a href="https://uniciv.com.br/" target="_blank">UNICIV</a>
9    <a href="images/sky.jpg" target="_parent">
10       <img src="sky-thumb.jpg" alt="Céu Azul">
11   </a>
12 </body>
```

Figura 19 - HTML: Definindo o atributo target da tag âncora "a"

Criação de âncoras de marcador

Você também pode criar âncoras de favoritos para permitir que os usuários pule para uma seção específica de uma página da web. Os marcadores são especialmente úteis se você tiver uma página da web muito longa.

A criação de marcadores é um processo de duas etapas: primeiro adicione o atributo `id` no elemento para onde deseja pular e, em seguida, use esse `id` precedido pelo sinal de cerquilha (#) como o valor do atributo `href` da tag `<a>`, conforme mostrado no exemplo a seguir:

```
6  <body>
7    <a href="#secaoA">Ir para a Seção A</a>
8    <h2 id="secaoA">Seção A</h2>
9    <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit...</p>
10 </body>
```

Figura 20 - HTML - Definindo marcadores de navegação na mesma página

Criação de links de download

Você também pode criar o link de download de arquivo exatamente da mesma maneira que inserir links de texto. Basta apontar o URL de destino para o arquivo que deseja disponibilizar para download.

No exemplo a seguir, criamos os links de download para arquivos ZIP, PDF e JPG.

```
6  <body>
7    <a href="downloads/test.zip">Download Zip</a>
8    <a href="downloads/masters.pdf">Download PDF</a>
9    <a href="downloads/sample.jpg">Download Image</a>
10 </body>
```

Figura 21 - HTML: Definindo links de downloads via tag <a>

Formatação de Texto HTML

Formatando Texto com HTML

HTML fornece várias tags que você pode usar para fazer com que algum texto em suas páginas da web apareça de forma diferente do texto normal, por exemplo, você pode usar a tag `<b>` para deixar o texto em negrito, tag `<i>` para tornar o texto itálico, tag `<mark>` para destacar o texto, tag `<code>` para exibir um fragmento de código de computador, tags `<ins>` e `<del>` para marcar inserções e exclusões editoriais e muito mais.

O exemplo a seguir demonstra as tags de formatação mais comumente usadas em ação. Agora, vamos tentar entender como essas tags funcionam basicamente:

```
6  <body>
7    <p>Isto é um texto <b>negrito</b>.</p>
8    <p>Isto é um texto <strong>muito importante!</strong>.</p>
9    <p>Isto é um texto <i>italico</i>.</p>
10   <p>Isto é um texto <em>emfatizado</em>.</p>
11   <p>Isto é um texto <mark>marcado</mark>.</p>
12   <p>Isto é um texto <code>de "codigo de computador"</code>.</p>
13   <p>Isto é um texto <small>diminuído/pequeno</small>.</p>
14   <p>Isto é um texto <sub>subscrito</sub> e <sup>supscrito</sup>.</p>
15   <p>Isto é um texto <del>deletado</del>.</p>
16   <p>Isto é um texto <ins>inserido/sublinhado</ins>.</p>
17 </body>
```

Figura 22 HTML: Formatando textos no HTML

E o resultado desse código seria exatamente assim:

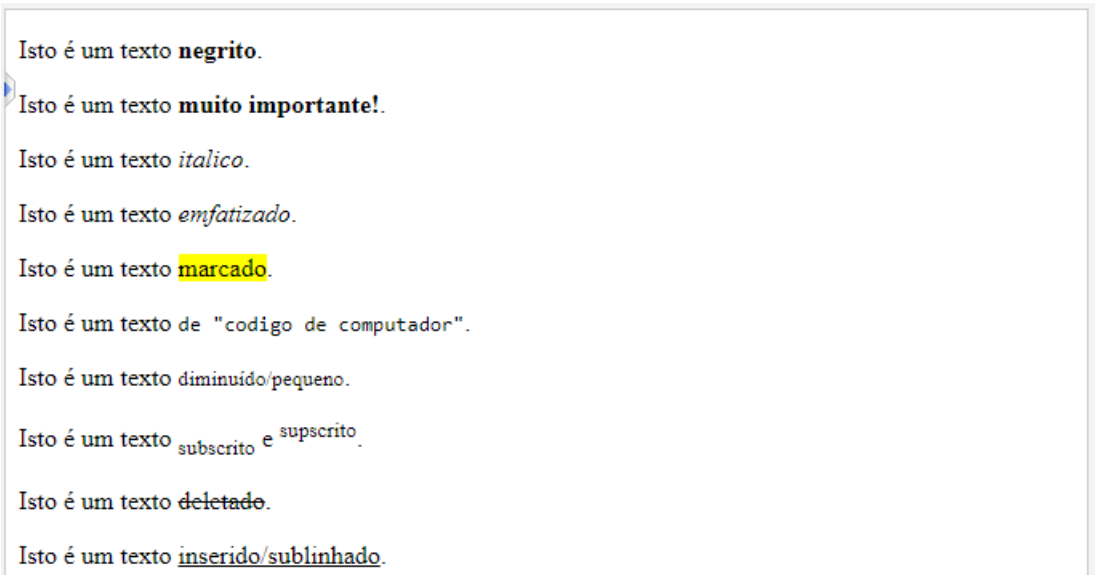

Figura 23 - HTML: Resultado da formatação de texto

Por padrão, a tag `<strong>` é normalmente renderizada no navegador como `<b>`, enquanto a tag `<em>` é renderizada como `<i>`. No entanto, há uma diferença no significado dessas tags.

Diferença entre e <b>

Ambas as tags `<strong>` e `<b>` renderizam o texto fechado em negrito por padrão, mas a tag `<strong>` indica que seu conteúdo tem grande importância, enquanto `<b>` é usada simplesmente para chamar a atenção do leitor, sem transmitir qualquer importância especial.

Diferença entre e <i>

Da mesma forma, as tags `<em>` e `<i>` renderizam o texto fechado em itálico por padrão, mas a tag `<em>` indica que seu conteúdo tem ênfase acentuada em comparação com o texto ao redor, enquanto `<i>` é usada para marcar o texto separado do texto normal por razões de legibilidade , como um termo técnico, uma frase idiomática de outro idioma, um pensamento, etc.

Formatando citações

Você pode formatar facilmente os blocos de cotação de outras fontes com a tag `<blockquote>` HTML .

Blockquotes geralmente são exibidos com as margens esquerda e direita recuadas, junto com um pouco de espaço extra adicionado acima e abaixo.

Para citações curtas inline, você pode usar a tag `<q>` HTML . A maioria dos navegadores exibe aspas em linha colocando o texto entre aspas.

Mostrando abreviações

Uma abreviatura é uma forma abreviada de uma palavra, frase ou nome.

Você pode usar a tag `<abbr>` para denotar uma abreviação. O atributo `title` é usado dentro desta tag para fornecer a expansão completa da abreviação, que é exibida pelos navegadores como uma dica de ferramenta quando o cursor do mouse passa sobre o elemento.

Marcando endereços de contato

As páginas da Web geralmente incluem ruas ou endereços postais. O HTML fornece uma marca especial utilizando `<address>` para representar informações de contato (físicas e / ou digitais) para uma pessoa, pessoas ou organização.

Essa tag deve ser usada idealmente para exibir informações de contato relacionadas ao próprio documento, como o autor do artigo. A maioria dos navegadores exibe um bloco de endereço em itálico.

Estilos HTML

Estilo de elementos HTML

O HTML é bastante limitado quando se trata da apresentação de uma página da web. Foi originalmente concebido como uma forma simples de apresentar informações. CSS (Cascading Style Sheets) foi introduzido em dezembro de 1996 pelo World Wide Web Consortium (W3C) para fornecer uma maneira melhor de estilizar os elementos HTML.

Com CSS, torna-se muito fácil especificar coisas como, tamanho e tipo de letra para as fontes, cores para o texto e fundos, alinhamento do texto e imagens, quantidade de espaço entre os elementos, borda e contornos para os elementos, e muito de outras propriedades de estilo.

Adicionando estilos a elementos HTML

As informações de estilo podem ser anexadas como um documento separado ou incorporadas ao próprio documento HTML. Esses são os três métodos de implementação de informações de estilo em um documento HTML.

- **Estilos embutidos (inline)**- usando o atributo `style` na tag de início HTML.
- **Estilo incorporado (embedded)** - usando o elemento `<style>` na seção principal do documento.
- **Folha de estilo externa** - usando o elemento `<link>`, apontando para arquivos CSS externos.

Estilos embutidos (inline)

Os estilos embutidos são usados para aplicar as regras de estilo exclusivas a um elemento, colocando as regras CSS diretamente na tag inicial. Ele pode ser anexado a um elemento usando o atributo `style`.

O atributo de estilo inclui uma série de propriedades CSS que são definidos por pares de valores. Cada `propriedade:valor` é separado por um ponto e vírgula (`;`), da mesma forma que você escreveria em uma folha de estilo incorporada ou externa. Mas precisa estar tudo em uma linha, ou seja, sem quebra de linha após o ponto-e-vírgula.

O exemplo a seguir demonstra como definir o `color` e `font-size` do texto:

```
<body>
  <h1 style="color:red; font-size:30px;">Isto é um heading</h1>
  <p style="color:green; font-size:18px;">Isto é um parágrafo.</p>
  <div style="color:green; font-size:18px;">Isto é um texto.</div>
</body>
```

Figura 24 - HTML: Aplicando CSS embutido (inline)

Usar os estilos embutidos é geralmente considerado uma prática ruim. Como as regras de estilo são incorporadas diretamente na tag html, isso faz com que a apresentação se misture com o conteúdo do documento, o que torna a atualização ou manutenção de um site muito ruim e difícil.

Folhas de estilo incorporadas (embedded)

As folhas de estilo incorporadas ou internas afetam apenas o documento em que estão incorporadas.

As folhas de estilo incorporadas são definidas na seção `<head>` de um documento HTML usando a tag `<style>`. Você pode definir qualquer número de elementos `<style>` dentro da seção `<head>`.

O exemplo a seguir demonstra como as regras de estilo são incorporadas em uma página da web.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Estilizando a página com CSS</title>
  <style>
    body { background-color: YellowGreen; }
    h1 { color: blue; }
    p { color: red; }
  </style>
</head>
<body>
```

Figura 25 - HTML: Estilizando a página com CSS embedded

Folhas de estilo externas

Uma folha de estilo externa é o formato de aplicação ideal quando o estilo é aplicado a muitas páginas.

Uma folha de estilo externa contém todas as regras de estilo em um documento separado que pode ser vinculado a qualquer documento HTML em seu site. Folhas de estilo externas são as mais flexíveis porque, com uma folha de estilo externa, você pode alterar a aparência de um site inteiro atualizando apenas um arquivo.

Você pode anexar folhas de estilo externas de duas maneiras - vinculando e importando.

Vinculando folhas de estilo externas

Uma folha de estilo externa pode ser vinculada a um documento HTML usando a tag `<link>`.

A tag `<link>` vai dentro da seção `<head>`, conforme mostrado aqui:

```
<head>
  <title>Estilizando a página com CSS</title>
    <link rel="stylesheet" href="css/style.css">
</head>
```

Figura 26 - HTML: Aplicando CSS via <link>

Importando folhas de estilo externas

A regra `@import` é outra maneira de carregar uma folha de estilo externa. A instrução `@import` instrui o navegador a carregar uma folha de estilo externa e usar seus estilos.

Você pode usá-lo de duas maneiras. A maneira mais simples é usá-lo dentro do elemento `<style>` em sua seção `<head>`. Observe que outras regras CSS ainda podem ser incluídas no elemento `<style>`.

```
<head>
  <title>Estilizando a página com CSS</title>
  <style>
      @import url("css/style.css");
      p {
          color: blue;
          font-size: 16px;
      }
  </style>
</head>
```

Figura 27 - HTML: Importando folhas de estilo via import

Da mesma forma, você pode usar a regra `@import` para importar uma folha de estilo dentro de outra folha de estilo.

```
<head>
  <title>Estilizando a página com CSS</title>
  <style>
      @import url("css/style.css");
      @import url("css/color.css");
      body {
          color: blue;
          font-size: 14px;
      }
      p {
          color: blue;
          font-size: 16px;
      }
  </style>
</head>
```

Figura 28 - HTML: Importando várias folhas de estilos via import

Imagens HTML

Inserindo imagens em páginas da web

As imagens melhoram a aparência visual das páginas da web, tornando-as mais interessantes e coloridas.

A tag `<img>` é usada para inserir imagens nos documentos HTML. É um elemento vazio e contém apenas atributos. A sintaxe da tag `<img>` pode ser fornecida com:

```
<img src="url" alt="Texto referência">
```

O exemplo a seguir insere três imagens na página da web:

```
18 <body>
19   <img src="kites.jpg" alt="Pipas voando">
20   <img src="sky.jpg" alt="Céu Azul">
21   <img src="balloons.jpg" alt="Balões">
22 </body>
```

Figura 29 - HTML: Inserindo imagens

Cada imagem deve conter pelo menos dois atributos: o atributo `src` e um atributo `alt`.

O atributo `src` informa ao navegador onde encontrar a imagem. Seu valor é a URL do arquivo de imagem.

Considerando que, o atributo `alt` fornece um texto alternativo para a imagem, se não estiver disponível ou não puder ser exibida por algum motivo. Seu valor deve ser um substituto significativo para a imagem.

Definindo a largura e altura de uma imagem

Os atributos `width` e `height` são usados para especificar respectivamente a largura e a altura de uma imagem.

Os valores desses atributos são interpretados em pixels por padrão.

```
18 <body>
19   <img src="kites.jpg" alt="Pipas voando" width="250" height="250">
20   <img src="sky.jpg" alt="Céu Azul" width="700" height="300">
21   <img src="balloons.jpg" alt="Balões" width="400" height="150">
22 </body>
```

Figura 30 - HTML: Definindo largura e altura de uma imagem

Você também pode usar o atributo `style` para especificar a largura e a altura das imagens. Impede que as folhas de estilo alterem o tamanho da imagem acidentalmente, pois o estilo embutido tem a prioridade mais alta.

```
18 <body>
19   <img src="kites.jpg" alt="Pipas voando" style="width: 250px; height:
   250px;">
20   <img src="sky.jpg" alt="Céu Azul" style="width: 700px; height: 300px;">
21   <img src="balloons.jpg" alt="Balões" style="width: 400px; height: 150px;">
22 </body>
```

Figura 31 - HTML: Definindo largura e altura de uma imagem através de CSS inline

Usando o elemento de imagem HTML5

Às vezes, aumentar ou diminuir uma imagem para caber em dispositivos diferentes (ou tamanhos de tela) não funciona conforme o esperado. Além disso, a redução da dimensão da imagem usando o atributo `width` e `height` não reduz o tamanho do arquivo original. Para resolver esses problemas, o HTML5 introduziu a tag `<picture>` que permite definir várias versões de uma imagem para diferentes tipos de dispositivos.

O elemento `<picture>` contém zero ou mais elementos `<source>`, cada um referindo-se a uma fonte de imagem diferente e um elemento `<img>` no final. Além disso, cada elemento `<source>` possui o atributo `media` que especifica uma condição de mídia (semelhante à consulta de mídia) que é usada pelo navegador para determinar quando uma fonte específica deve ser usada. Vamos tentar um exemplo:

```
6  <body>
7    <picture>
8        <source media="(min-width: 1000px)" srcset="logo-large.png">
9        <source media="(max-width: 500px)" srcset="logo-small.png">
10       <img src="logo-default.png" alt="Minha logo">
11   </picture>
12 </body>
```

Figura 32 - HTML: Exibindo imagens de forma responsiva com Picture

> **Nota:** O navegador avaliará cada elemento <source> filho e escolherá a melhor correspondência entre eles. Se nenhuma correspondência for encontrada, o atributo "src" do elemento <img> é usado. Além disso, a tag <img> deve ser especificada como **o último filho** do elemento <picture>.

Trabalhando com Mapas de Imagens

Um mapa de imagem permite definir pontos de acesso em uma imagem que atua como um hiperlink.

A ideia básica por trás da criação de um mapa de imagem é fornecer uma maneira fácil de vincular várias partes de uma imagem sem dividi-la em arquivos de imagem separados. Por exemplo, um mapa-múndi pode ter cada país vinculado a mais informações sobre aquele país.

Vamos experimentar um exemplo simples para entender como ele realmente funciona:

```
7    <img src="objetos.png" usemap="#objetos" alt="Objetos Geometricos">
8    <map name="objetos">
9        <area shape="circle" coords="137,231,71" href="circulo.html"
alt="Circulo">
10       <area shape="poly" coords="363,146,273,302,452,300"
href="poligono.html" alt="Poligono">
11       <area shape="rect" coords="520,160,641,302" href="retangulo.html"
alt="Retangulo">
12   </map>
```

Figura 33 - HTML: Trabalhando com Mapas de Imagens

O atributo `name` da tag `<map>` é usado para fazer referência ao mapa da tag `<img>` usando seu atributo `usemap`. A tag `<area>` é usada dentro do elemento `<map>` para definir as áreas clicáveis em uma imagem. Você pode definir qualquer número de áreas clicáveis em uma imagem.

Tabelas HTML

Criação de tabelas em HTML

A tabela HTML permite que você organize os dados em linhas e colunas. Eles são comumente usados para exibir dados tabulares como listas de produtos, detalhes do cliente, relatórios financeiros e assim por diante.

Você pode criar uma tabela usando o elemento `<table>`. Dentro do elemento `<table>`, você pode usar os elementos `<tr>` para criar linhas e, para criar colunas dentro de uma linha, você pode usar os elementos `<td>`. Você também pode definir uma célula como cabeçalho de um grupo de células da tabela usando o elemento `<th>`.

O exemplo a seguir demonstra a estrutura mais básica de uma tabela.

```
7   <table>
8       <tr>
9           <th>No.</th>
10          <th>Nome</th>
11          <th>Idade</th>
12      </tr>
13      <tr>
14          <td>1</td>
15          <td>Peter Parker</td>
16          <td>16</td>
17      </tr>
18      <tr>
19          <td>2</td>
20          <td>Clark Kent</td>
21          <td>34</td>
22      </tr>
23  </table>
```

Figura 34 - HTML: Criando uma tabela

As tabelas não têm bordas por padrão. Você pode usar a propriedade CSS `border` para adicionar bordas às tabelas.
Além disso, as células da tabela são dimensionadas apenas grandes o suficiente para caber no conteúdo por padrão. Para adicionar mais espaço ao redor do conteúdo nas células da tabela, você pode usar a propriedade CSS `padding`.

As regras de estilo a seguir adicionam uma borda de 1 pixel à tabela e 10 pixels de preenchimento às células.

```
<style>
  table, th, td {
      border: 1px solid black;
  }
  th, td {
      padding: 10px;
  }
</style>
```

Figura 35 - HTML: Estilizando tabelas

Por padrão, as bordas ao redor da tabela e suas células são separadas umas das outras. Mas você pode agrupá-los em um usando a propriedade `border-collapse` no elemento `<table>`.

Além disso, o texto dentro dos elementos `<th>` é exibido em negrito, alinhado horizontalmente no centro da célula por padrão. Para alterar o alinhamento padrão, você pode usar a propriedade CSS `text-align`.

As regras de estilo a seguir reduzem as bordas da tabela e alinham o texto do cabeçalho da tabela à esquerda.

```
table {
  border-collapse: collapse;
}
th {
    text-align: left;
}
```

Figura 36 - HTML - Ajustando o layout de uma tabela

Abrangendo várias linhas e colunas

A expansão permite que você estenda as linhas e colunas da tabela em várias outras linhas e colunas.

Normalmente, uma célula da tabela não pode passar para o espaço abaixo ou acima de outra célula da tabela. Porém, você pode usar os atributos `rowspan` ou `colspan` para abranger várias linhas ou colunas em uma tabela.

Da mesma forma, você pode usar o atributo `rowspan` para criar uma célula que se estende por mais de uma linha.

Adicionar legendas às tabelas

Você pode especificar uma legenda (ou título) para suas tabelas usando o elemento `<caption>`.

O elemento `<caption>` deve ser colocado diretamente após a tag `<table>` de abertura. Por padrão, a legenda aparece no topo da tabela, mas você pode alterar sua posição usando a propriedade CSS `caption-side`.

Definindo um cabeçalho, corpo e rodapé de tabela

O HTML fornece uma série de tags `<thead>`, `<tbody>` e `<tfoot>` que ajuda você a criar uma tabela mais estruturada, definindo as regiões de cabeçalho, corpo e rodapé, respectivamente.

Vejamos como fica todos estes atributos quando aplicados:

```
<table>
      <caption>Meus heróis preferidos</caption>
    <thead>
        <tr>
            <th>Nome</th>
            <th>Universo</th>
        </tr>
    </thead>
    <tbody>
        <tr>
            <td rowspan="2">Marvel</td>
            <td>Spider Man</td>
        </tr>
        <tr>
            <td>Iron Man</td>
        </tr>
        <tr>
            <td rowspan="2">DC</td>
            <td>Batman</td>
        </tr>
        <tr>
            <td>Superman</td>
        </tr>
    </tbody>
    <tfoot>
        <tr>
            <th colspan="2">Total: 4 herois</th>
        </tr>
    </tfoot>
</table>
```

Figura 37 - HTML: Aplicando atributos na tabela para controle de disposição de dados

Listas HTML

Trabalho com listas de HTML

Listas HTML são usadas para apresentar lista de informações de forma bem formada e semântica. Existem três tipos diferentes de lista em HTML e cada um tem um propósito e significado específicos.

- **Lista não ordenada** - usada para criar uma lista de itens relacionados, sem uma ordem específica.
- **Lista ordenada** - usada para criar uma lista de itens relacionados, em uma ordem específica.
- **Lista de descrição** - usada para criar uma lista de termos e suas descrições.

Listas HTML não ordenadas

Uma lista não ordenada criada usando o elemento `<ul>`, e cada item da lista começa com o elemento `<li>`.

Os itens da lista em listas não ordenadas são marcados com marcadores.

```
<ul>
    <li>Chocolate</li>
    <li>Morangos</li>
    <li>Chantilly</li>
</ul>
```

Figura 38 - HTML: Trabalhando com listas não ordenadas

Você também pode alterar o tipo de marcador em sua lista não ordenada usando a propriedade CSS `list-style-type`.

```
ul {
    list-style-type: square;
}
```

Figura 39 - HTML: Alterando o tipo do marcador de uma lista não ordenada

Listas ordenadas em HTML

Uma lista ordenada criada usando o elemento `<ol>`, e cada item da lista começa com o elemento `<li>`. Listas ordenadas são usadas quando a ordem dos itens da lista é importante.

Os itens da lista em uma lista ordenada são marcados com números.

```
<ol>
    <li>Afivele seu cinto de segurança.</li>
    <li>Dê partida em seu veiculo.</li>
    <li>Olhe no retrovisor e se estiver livre, avançe....</li>
</ol>
```

Figura 40 - HTML: Trabalhando com listas ordenadas

A numeração de itens em uma lista ordenada normalmente começa com 1. No entanto, se você quiser alterar isso, pode usar o atributo `start` no elemento `<ol>.`

Como a lista não ordenada, você também pode usar a propriedade CSS `list-style-type` para alterar o tipo de numeração em uma lista ordenada.

A seguinte regra de estilo altera o tipo de marcador para números romanos.

```
ol {
    list-style-type: upper-roman;
}
```

Figura 41 - HTML: Alterando o tipo do marcador numérico de uma lista ordenada

Listas de descrição de HTML

Uma lista de descrição é uma lista de itens com uma descrição ou definição de cada item.

A lista de descrição é criada usando o elemento `<dl>`. O elemento `<dl>` é usado em conjunto com o elemento `<dt>` que especifica um termo e o elemento `<dd>` que especifica a definição do termo.

Os navegadores geralmente exibem as listas de definições colocando os termos e definições em linhas separadas, onde as definições dos termos são ligeiramente recuadas

```
<dl>
    <dt>Pão</dt>
    <dd>Assado e feito com farinha integral.</dd>
    <dt>Café</dt>
    <dd>Forte e passado com grãos bem tostados.</dd>
</dl>
```

Figura 42 - HTML: Lista de descrição

Formulários HTML

O que é um formulário HTML?

Os formulários HTML são necessários para coletar diferentes tipos de entradas do usuário, como detalhes de contato como nome, endereço de e-mail, números de telefone ou detalhes como informações de cartão de crédito, etc.

Os formulários contêm elementos especiais chamados de controles como caixa de entrada, caixas de seleção, botões de rádio, botões de envio, etc. Os usuários geralmente preenchem um formulário modificando seus controles, por exemplo, inserindo texto, selecionando itens, etc. e enviando este formulário a um servidor da web para processamento posterior.

A tag `<form>` é usada para criar um formulário HTML.

Elemento de entrada

Este é o elemento mais comumente usado em formulários HTML.

Ele permite que você especifique vários tipos de campos de entrada do usuário, dependendo do atributo `type`. Um elemento de entrada pode ser do tipo *campo de texto* , *campo de senha*, *caixa*, *botão de rádio*, *botão enviar*, *botão de reset*, *arquivo*, *caixa de seleção* , bem como vários tipos de entrada de novas introduzidas no HTML5.

Os tipos de entrada usados com mais frequência são descritos a seguir.

Campos de Texto

Os campos de texto são áreas de uma linha que permitem ao usuário inserir texto.

Os controles de entrada de texto de linha única são criados usando um elemento `<input>`, cujo atributo `type` tem um valor de `text`.

Campo de senha

Os campos de senha são semelhantes aos campos de texto. A única diferença é; os caracteres em um campo de senha são mascarados, ou seja, são exibidos como asteriscos ou pontos. Isso evita que outra pessoa leia a senha na tela. Este também é um controle de entrada de texto de linha única criado usando um elemento `<input>` cujo atributo `type` tem um valor de `password`.

```
<form>
    <label>Username: <input type="text"></label>
    <label>Password: <input type="password"></label>
    <input type="submit" value="Submit">
</form>
```

Figura 43 - HTML: Exemplo de Formulário com input text e password

Botões do rádio

Os botões de rádio são usados para permitir que o usuário selecione exatamente uma opção de um conjunto predefinido de opções. Ele é criado usando um elemento `<input>` cujo atributo `type` tem um valor de `radio`.

```html
<form>
    <input type="radio" name="genero" id="masculino">
    <label for="masculino">Masculino</label>
    <input type="radio" name="genero" id="feminino">
    <label for="feminino">Feminino</label>
    <input type="radio" name="genero" id="outro">
    <label for="outro">Outro</label>
</form>
```

Figura 44 - HTML: Exemplo de Formulário com radio button

Caixas de seleção

As caixas de seleção permitem que o usuário selecione uma ou mais opções de um conjunto predefinido de opções. Ele é criado usando um elemento `<input>` cujo atributo `type` tem um valor de `checkbox`.

```html
<form>
    <input type="checkbox" name="esportes" id="futebol">
    <label for="futebol">Futebol</label>
    <input type="checkbox" name="esportes" id="basquete">
    <label for="basquete">Basquete</label>
    <input type="checkbox" name="esportes" id="volei">
    <label for="volei">Volei</label>
</form>
```

Figura 45 - HTML: Exemplo de Formulário com checkbox

Caixa de seleção de arquivo

Os campos do arquivo permitem que o usuário procure um arquivo local e envie-o como um anexo com os dados do formulário. Navegadores da Web como Google Chrome e Firefox renderizam um campo de entrada de seleção de arquivo com um botão Procurar que permite ao usuário navegar no disco rígido local e selecionar um arquivo.

Isso também é criado usando um elemento `<input>`, cujo valor do atributo `type` é definido como `file`.

```html
<form>
    <label for="file-select">Upload:</label>
    <input type="file" name="upload" id="file-select">
</form>
```

Figura 46 - HTML: Exemplo de Formulário com file

Textarea

Textarea é um controle de entrada de texto de várias linhas que permite ao usuário inserir mais de uma linha de texto. Os controles de entrada de texto multilinha são criados usando um elemento `<textarea>`.

```html
<form>
    <label for="endereco">Endereço:</label>
    <textarea rows="3" cols="30" name="endereco" id="endereco"></textarea>
</form>
```

Figura 47 - HTML: Exemplo de Formulário com textarea

Caixas de Seleção

Uma caixa de seleção é uma lista suspensa de opções que permite ao usuário selecionar uma ou mais opções em uma lista suspensa de opções. A caixa de seleção é criada usando o elemento `<select>` e o elemento `<option>`.

Os `<option>`elementos dentro do elemento `<select>` definem cada item da lista.

```html
<form>
  <label for="cidade">Cidades:</label>
  <select name="cidade" id="cidade">
      <option value="maringa">Maringa</option>
      <option value="londrina">Londrina</option>
      <option value="curitiba">Curitiba</option>
  </select>
</form>
```

Figura 48 - HTML: Exemplo de Formulário com select

Botões de envio e redefinição

Um botão de envio é usado para enviar os dados do formulário para um servidor web. Quando o botão enviar é clicado, os dados do formulário são enviados para o arquivo especificado no atributo `action` do formulário para processar os dados enviados.

Um botão de redefinição redefine todo o controle de formulários para os valores padrão. Experimente o exemplo a seguir digitando seu nome no campo de texto e clique no botão enviar para vê-lo em ação.

```html
<form action="salvar.php" method="post">
    <label for="nome">Nome:</label>
    <input type="text" name="nome" id="nome">
    <input type="submit" value="Enviar">
    <input type="reset" value="Limpar">
</form>
```

Figura 49 - HTML: Exemplo de botões com ações de Enviar e Limpar formulários

Controles de formulário de agrupamento

Você também agrupa controles e rótulos logicamente relacionados em um formulário da web usando o elemento `<legend>`. O agrupamento de controles de formulário em categorias torna mais fácil para os usuários localizar um controle, o que torna o formulário mais amigável.

Atributos de formulário usados com frequência

A tabela a seguir lista os atributos dos elementos de formulário usados com mais frequência:

Tabela 2 - HTML: Atritos de formulários usados com frequência

`name`	Especifica o nome do formulário.
`action`	Especifica a URL do programa ou script no servidor web que será usado para processar as informações enviadas por meio do formulário.
`method`	Especifica o método HTTP usado para enviar os dados ao servidor da web pelo navegador. O valor pode ser `get`(o padrão) e `post`.
`target`	Especifica onde exibir a resposta recebida após o envio do formulário. Os valores possíveis são `_blank`, `_self`, `_parente` `_top`.
`enctype`	Especifica como os dados do formulário devem ser codificados ao enviar o formulário ao servidor. Aplicável apenas quando o valor do atributo `method` é `post`.
`name`	Especifica o nome do formulário.
`action`	Especifica a URL do programa ou script no servidor web que será usado para processar as informações enviadas por meio do formulário.
`method`	Especifica o método HTTP usado para enviar os dados ao servidor da web pelo navegador. O valor pode ser `get`(o padrão) e `post`.
`target`	Especifica onde exibir a resposta recebida após o envio do formulário. Os valores possíveis são `_blank`, `_self`, `_parente` `_top`.

`enctype`	Especifica como os dados do formulário devem ser codificados ao enviar o formulário ao servidor. Aplicável apenas quando o valor do `method`atributo é `post`.
`name`	Especifica o nome do formulário.
`action`	Especifica a URL do programa ou script no servidor web que será usado para processar as informações enviadas por meio do formulário.
`method`	Especifica o método HTTP usado para enviar os dados ao servidor da web pelo navegador. O valor pode ser `get`(o padrão) e `post`.
`target`	Especifica onde exibir a resposta recebida após o envio do formulário. Os valores possíveis são `_blank`, `_self`, `_parente` `_top`.

HTML Meta

Definindo Metadados

As tags `<meta>` são normalmente usadas para fornecer metadados estruturados, como *palavras-chave de* um documento, *descrição*, *nome do autor*, *codificação de caracteres* e outros metadados. Qualquer número de metatags pode ser colocado dentro da seção `<head>` de um documento HTML ou XHTML.

Os metadados não serão exibidos na página da web, mas serão analisáveis por máquina e podem ser usados pelos navegadores, mecanismos de pesquisa como o Google ou outros serviços da web.

A seção a seguir descreve o uso de meta tags para vários fins.

Declaração de codificação de caracteres em HTML

Meta tag normalmente usada para declarar a codificação de caracteres dentro de um documento HTML.

Definindo o autor de um documento

Você também pode usar a metatag para definir claramente quem é o autor ou criador da página da web.

O autor pode ser um indivíduo, a empresa como um todo ou um terceiro.

Palavras-chave e descrição para motores de busca

Alguns mecanismos de pesquisa usam metadados, especialmente palavras-chave e descrições para indexar páginas da web; no entanto, isso pode não ser necessariamente verdade. Palavras-chave que dão peso extra às palavras-chave e à descrição de um documento fornecem uma breve sinopse da página.

Configurando a janela de visualização (Viewport) para dispositivos móveis

Você pode usar a meta tag viewport para exibir as páginas da web corretamente em dispositivos móveis.

Sem uma metatag viewport, os navegadores móveis renderizam as páginas da web em larguras de tela de desktop típicas e, em seguida, diminuem a escala para caber na tela do celular. Como resultado, é necessário apertar e aplicar zoom para visualizar a página da web corretamente em dispositivos móveis, o que é muito inconveniente.

A metatag viewport permite definir o melhor tamanho da janela de visualização e os limites de escala para visualizar as páginas da web em dispositivos móveis.

O par de valores-chave `width=device-width` dentro do atributo `content` define a largura da janela de visualização igual à largura da tela do dispositivo, enquanto `initial-scale=1` define a escala inicial ou o nível de zoom como 100% quando a página é carregada pela primeira vez pelo navegador.

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <title>Declarando meta-tags</title>
5      <meta charset="utf-8">
6      <meta name="author" content="Douglas Mariano | UNICIV">
7      <meta name="keywords" content="HTML, CSS, javaScript">
8      <meta name="description" content="Tutorial com referências simples de entender
   em HTML, CSS, javaScript e muito mais...">
9      <meta name="viewport" content="width=device-width, initial-scale=1">
10
11 </head>
12 <body>
13     <h1>Olá Mundo!</h1>
14 </body>
15 </html>
```

Figura 50 - HTML: Aplicando meta-tags ao HEAD do HTML

Introdução CSS

CSS é uma linguagem de folha de estilo (Cascading Style Sheets) usada para descrever a apresentação, ou seja, o layout e a formatação das páginas da web.

Antes do CSS, quase todos os atributos de apresentação de documentos HTML estavam contidos na marcação HTML (especificamente dentro das tags HTML).
Todas as cores da fonte, estilos de fundo, alinhamentos de elementos, bordas e tamanhos tinham que ser explicitamente descritos no HTML.

Como resultado, o desenvolvimento de grandes sites tornou-se um processo longo e caro, uma vez que as informações de estilo foram adicionadas repetidamente a cada página do site.

Para resolver este problema, o CSS foi introduzido em 1996 pelo W3C para permitir a separação de apresentação e conteúdo. Agora, os web designers podiam mover as informações de formatação das páginas da web para uma folha de estilo separada, o que resultou em uma marcação HTML consideravelmente mais simples e de melhor manutenção.

CSS3 é a versão mais recente da especificação CSS e adiciona vários novos recursos de estilo e melhorias para aprimorar os recursos de apresentação na web.

O que você pode fazer com CSS

Existem muitas coisas que você pode fazer com CSS.

- Você pode aplicar facilmente as mesmas regras de estilo em vários elementos.
- Você pode controlar a apresentação de várias páginas de um site com uma única folha de estilo.
- Você pode apresentar a mesma página de maneira diferente em dispositivos diferentes.
- Você pode definir o estilo de estados dinâmicos de elementos como pairar, foco, etc. que não são possíveis de outra forma.
- Você pode alterar a posição de um elemento em uma página da web sem alterar a marcação.
- Você pode alterar a exibição de elementos HTML existentes.
- Você pode transformar elementos como dimensionar, girar, inclinar, etc. em espaço 2D ou 3D.
- Você pode criar animações e efeitos de transição sem usar JavaScript.
- Você pode criar uma versão amigável para impressão de suas páginas da web.

Vantagens de usar CSS

A maior vantagem do CSS é que ele permite separar o estilo e o layout do conteúdo do documento. Aqui estão mais algumas vantagens, por que se deve começar a usar CSS?

- **CSS economiza muito tempo** - CSS oferece muita flexibilidade para definir as propriedades de estilo de um elemento. Você pode escrever CSS uma vez; e então o mesmo código pode ser aplicado aos grupos de elementos HTML e também pode ser reutilizado em várias páginas HTML.
- **Manutenção fácil** - CSS fornece um meio fácil de atualizar a formatação dos documentos e de manter a consistência em vários documentos. Porque o conteúdo de

todo o conjunto de páginas da web pode ser facilmente controlado usando uma ou mais folhas de estilo.

- **Páginas carregam mais rápido** - CSS permite que várias páginas compartilhem as informações de formatação, o que reduz a complexidade e a repetição no conteúdo estrutural dos documentos. Reduz significativamente o tamanho da transferência do arquivo, o que resulta em um carregamento mais rápido da página.
- **Estilos superiores para HTML** - CSS tem recursos de apresentação muito mais amplos do que HTML e fornece um controle muito melhor sobre o layout de suas páginas da web. Assim, você pode dar uma aparência muito melhor às suas páginas da web em comparação aos elementos e atributos de apresentação HTML.
- **Compatibilidade com vários dispositivos** - CSS também permite que páginas da web sejam otimizadas para mais de um tipo de dispositivo ou mídia. Usando CSS, o mesmo documento HTML pode ser apresentado em diferentes estilos de visualização para diferentes dispositivos de renderização, como desktop, telefones celulares, etc.

Seletores CSS

O que é Seletor?

Um seletor CSS é um padrão para combinar os elementos em uma página da web. As regras de estilo associadas a esse seletor serão aplicadas aos elementos que correspondem ao padrão do seletor.

Os seletores são um dos aspectos mais importantes do CSS, pois permitem que você direcione elementos específicos em sua página da web de várias maneiras para que possam ser estilizados.

Vários tipos de seletores estão disponíveis em CSS, vamos dar uma olhada neles:

Seletor Universal

O seletor universal, indicado por um asterisco (*), corresponde a cada elemento da página.

O seletor universal pode ser omitido se outras condições existirem no elemento. Este seletor é frequentemente usado para remover as margens e preenchimentos padrão dos elementos para fins de teste rápido.

```
1  * {
2      margin: 0;
3      padding: 0;
4  }
```

Figura 51 - CSS: Seletor universal - reset de página

Seletores de tipo de elemento

Um seletor de tipo de elemento corresponde a todas as instâncias do elemento no documento com o nome do tipo de elemento correspondente.

```
1  p {
2      color: blue;
3  }
```

Figura 52 - CSS: Seletor de elemento ou tag

Seletores de Id

O seletor de id é usado para definir regras de estilo para um elemento *único* ou *exclusivo*.

O seletor de id é definido com um sinal de hashtag/cerquilha (`#`) imediatamente seguido pelo valor de id.

```
1  #error {
2      color: red;
3  }
```

Figura 53 - CSS: Seletor de ID

Seletores de classe

Os seletores de classe podem ser usados para selecionar qualquer elemento HTML que tenha um atributo `class`. Todos os elementos que possuem essa classe serão formatados de acordo com a regra definida.

O seletor de classe é definido com um sinal de ponto (`.`) seguido imediatamente pelo valor da classe.

```
1  .blue {
2      color: blue;
3  }
```

Figura 54 - CSS: Seletor de Classe

Seletores de descendentes

Você pode usar esses seletores quando precisar selecionar um elemento que é descendente de outro elemento, por exemplo, se desejar direcionar apenas os links que estão contidos em uma lista não ordenada, em vez de direcionar todos os elementos do link.

```
1  ul.menu li a {
2      text-decoration: none;
3  }
4  h1 em {
5      color: green;
6  }
```

Figura 55 - CSS: Seletor de descendência

As regras de estilo dentro do seletor são `ul.menu li a` aplicadas apenas aos elementos `<a>` contidos em um elemento `<ul>` que possui a classe `.menu` e não têm efeito em outros links dentro do documento.

Da mesma forma, as regras de estilo dentro do seletor `h1 em` serão aplicadas apenas aos elementos `<em>` contidos dentro do elemento `<h1>` e não têm efeito em outros elementos `<em>`.

Seletores filhos

Um seletor filho é usado para selecionar apenas os elementos que são filhos diretos de algum elemento.

Um seletor filho é composto de dois ou mais seletores separados por um símbolo de maior (`>`). Você pode usar este seletor, por exemplo, para selecionar o primeiro nível de elementos de lista dentro de uma lista aninhada que tem mais de um nível.

```
1  ul > li {
2      list-style: square;
3  }
4  ul > li ol {
5      list-style: none;
6  }
```

Figura 56 - CSS: Exemplo de seletor filho

Seletores de irmãos adjacentes

Os seletores de irmãos adjacentes (Sibling) podem ser usados para selecionar elementos irmãos (ou seja, elementos no mesmo nível). Este seletor tem a sintaxe como: E1 + E2, onde E2 é o destino do seletor.

O seletor `h1 + p` no exemplo a seguir selecionará os elementos `<p>` apenas se os elementos `<h1>` e `<p>` compartilharem o mesmo pai na árvore do documento e `<h1>` for imediatamente anterior ao elemento `<p>`. Isso significa que apenas os parágrafos que vêm imediatamente após cada título `<h1>` terão as regras de estilo associadas.

```
1  h1 + p {
2      color: blue;
3      font-size: 18px;
4  }
5  ul.task + p {
6      color: #f0f;
7      text-indent: 30px;
8  }
```

Figura 57 - CSS: Seletor de irmãos adjacentes (sibling adjacent)

Seletores gerais de irmãos

O seletor geral de irmãos é semelhante ao seletor de irmãos adjacentes (E1 + E2), mas é menos estrito. Um seletor irmão geral é composto de dois seletores simples separados pelo caractere `~` (til). Pode ser escrito como: E1 ~ E2, onde E2 é o alvo do seletor.

O seletor `h1 ~ p` no exemplo abaixo selecionará todos os elementos `<p>` que precedem o elemento `<h1>`, onde todos os elementos compartilham o mesmo pai na árvore do documento.

```css
h1 ~ p {
    color: blue;
    font-size: 18px;
}
ul.task ~ p {
    color: #f0f;
    text-indent: 30px;
}
```

Figura 58 - CSS: Seletor geral de irmãos (general sibling)

Seletores de agrupamento

Frequentemente, vários seletores em uma folha de estilo compartilham as mesmas declarações de regras de estilo. Você pode agrupá-los em uma lista separada por vírgulas para minimizar o código em sua folha de estilo. Também evita que você repita as mesmas regras de estilo indefinidamente.

```css
h1, h2, h3 {
    font-weight: normal;
}
h1 {
    font-size: 36px;
}
h2 {
    font-size: 28px;
}
h3 {
    font-size: 22px;
}
```

Figura 59 - CSS: Agrupamento de seletores

CSS Color

Configurando a propriedade da cor

A propriedade `color` define a cor do texto (cor de primeiro plano em geral) de um elemento.

Por exemplo, a propriedade `color` especificada no seletor `body` define a cor do texto padrão para toda a página.

```css
body {
    color: #ff5722;
}
```

Figura 60 - CSS: Definindo uma cor

Definindo Valores de Cor

As cores em CSS são mais frequentemente especificadas nos seguintes formatos:

- uma palavra-chave de cor - como "vermelho", "verde", "azul", "transparente" etc.
- um valor HEX - como "# ff0000", "# 00ff00", etc.

- um valor RGB - como "rgb (255, 0, 0)"

CSS3 introduziu vários outros formatos de cores, como HSL, HSLA e RGBA, que também suportam transparência alfa.

Por enquanto, vamos nos ater aos métodos básicos de definição dos valores das cores.

Palavras-chave de cor

CSS define algumas palavras-chave de cores que permitem que você especifique os valores das cores de uma maneira fácil.

Essas palavras-chave de cores básicas são: **aqua, black, blue, fuchsia, gray, green, lime, maroon, navy, olive, purple, red, silver, teal, white** e **yellow**. Os nomes das cores não diferenciam maiúsculas de minúsculas e devem sempre estar em inglês.

Os navegadores modernos, no entanto, praticamente suportam muito mais nomes de cores do que os definidos no padrão CSS, mas para ficar mais seguro, você deve usar valores de cor hexadecimais.

Valores de cor HEX

Hex (abreviação de hexadecimal) é de longe o método mais comumente usado para definir cores na web.

Hex representar as cores usando um código de seis dígitos, precedido por um caractere, como `#rrggbb`, em que `rr`, `gg` e `bb` representa o componente vermelho, verde e azul da cor respectivamente.

O valor de cada componente pode variar de 00 (sem cor) e FF (cor total) em notação hexadecimal ou 0 e 255 em notação decimal equivalente. Portanto, `#ffffff` representa a cor branca e `#000000` representa a cor preta.

Valores de cor RGB

As cores podem ser definidas no modelo RGB (vermelho, verde e azul) usando a notação funcional `rgb()`.

A função `rgb()` aceita três valores separados por vírgula, que especificam a quantidade de componente vermelho, verde e azul da cor. Esses valores são normalmente especificados como inteiros entre 0 e 255, onde 0 representa *nenhuma cor* e 255 representa *a cor total ou máxima.*

Aplicação da propriedade da cor nas bordas e contornos

A propriedade `color` não é apenas para conteúdo de texto, mas para qualquer coisa em primeiro plano que tenha um valor de cor. Por exemplo, se o valor `border-color` ou `outline-color` não foi definido explicitamente para o elemento, o valor da cor será usado.

Fundo CSS – Background

Configurando propriedades de fundo

O plano de fundo desempenha um papel importante na apresentação visual de uma página da web.

CSS fornece várias propriedades para definir o estilo do plano de fundo de um elemento, incluindo colorir o plano de fundo, colocar imagens no plano de fundo e gerenciar seu posicionamento, etc.

As propriedades de fundo são `background-color`, `background-image`, `background-repeat`, `background-attachmente` `background-position`.

Cor de fundo – (color)

A propriedade `background-color` é usada para definir a cor de fundo de um elemento.

```
1  body {
2      background-color: #f0e68c;
3  }
```

Figura 61 - CSS: Aplicando cor de fundo

Imagem de fundo – (image)

A propriedade `background-image` define uma imagem como plano de fundo de um elemento HTML.

```
1  body {
2      background-image: url("images/tile.png");
3  }
```

Figura 62 - CSS: Aplicando uma imagem como fundo

Repetição de imagem de fundo – (repeat)

A propriedade `background-repeat` permite controlar como uma imagem de plano de fundo é repetida ou ladrilhada no plano de fundo de um elemento. Você pode definir uma imagem de fundo para repetir verticalmente (eixo y), horizontalmente (eixo x), em ambas as direções ou em nenhuma direção.

Vamos experimentar o exemplo a seguir, que demonstra como definir o fundo gradiente para uma página da web, repetindo a imagem fatiada horizontalmente ao longo do eixo x.

```
1  body {
2      background-image: url("images/gradient.png");
3      background-repeat: repeat-x;
4  }
```

Figura 63 - CSS: Repetindo imagem de fundo no eixo x

Da mesma forma, você pode usar o valor `repeat-y` para repetir a imagem de plano de fundo verticalmente ao longo do eixo y ou o valor `no-repeat` para evitar a repetição por completo.

Vamos dar uma olhada na ilustração a seguir para entender como essa propriedade realmente funciona.

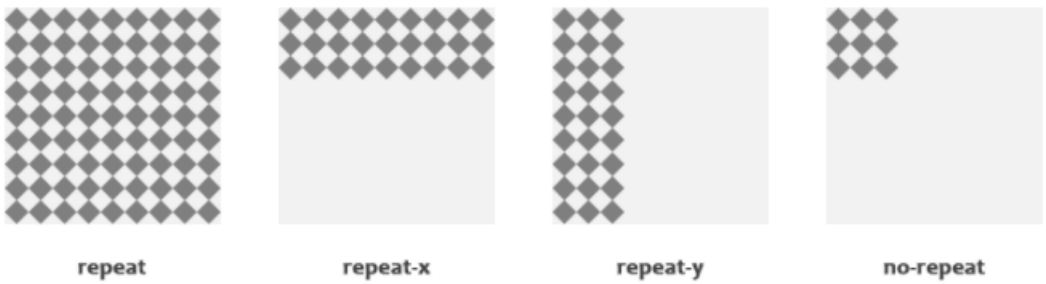

Figura 64 - CSS: Exemplo geral da aplicação do repeat

Posição de fundo – (position)

A propriedade `background-position` é usada para controlar a posição da imagem de fundo.

Se nenhuma posição de fundo foi especificada, a imagem de fundo é colocada na posição superior esquerda padrão do elemento, ou seja `(0,0).`

No exemplo a seguir, a imagem de fundo é posicionada no canto superior direito.

```
1  body {
2      background-image: url("images/robot.png");
3      background-repeat: no-repeat;
4      background-position: right top;
5  }
```

Figura 65 - CSS: Definindo posição de background

Além de palavras-chave, você também pode usar valores de porcentagem ou comprimento, como `px` ou `em` para esta propriedade.

Vamos dar uma olhada na ilustração a seguir para entender como essa propriedade realmente funciona.

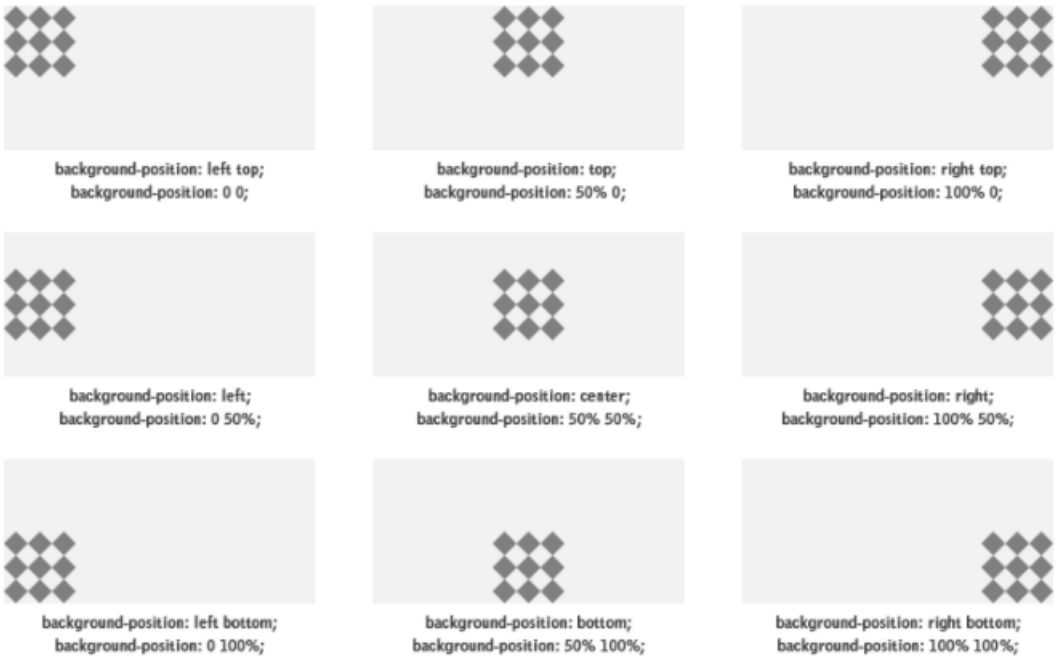

Figura 66 - CSS: Definindo background-position de formas diferentes

Anexo de Fundo – (attachment)

A propriedade `background-attachment` determina se a imagem de fundo é fixa em relação à janela de visualização ou rola junto com o bloco que o contém.

```
1  body {
2      background-image: url("images/bell.png");
3      background-repeat: no-repeat;
4      background-attachment: fixed;
5  }
```

Figura 67 - CSS: Determinando comportamento da imagem de fundo

A propriedade curta de definição de fundo – (shorthand)

Como você pode ver nos exemplos acima, há muitas propriedades a serem consideradas ao lidar com os planos de fundo. No entanto, também é possível especificar todas essas propriedades em uma única propriedade para encurtar o código ou evitar digitação extra. Isso é chamado de propriedade abreviada.

A propriedade `background` é um atalho para definir todas as propriedades de fundo individuais, ou seja, `background-color`, `background-image`, `background-repeat`, `background-attachment` e o `background-position` de uma só vez. Ou seja, você pode transformar isso:

```
1  body {
2      background-color: #f0e68c;
3      background-image: url("images/smiley.png");
4      background-repeat: no-repeat;
5      background-attachment: fixed;
6      background-position: 250px 25px;
7  }
```

Figura 68 - CSS: Propriedades de fundo

Em algo mais simples, como isto:

```
body {
    background: #f0e68c url("images/smiley.png") no-repeat fixed 250px 25px;
}
```

Figura 69 - CSS: Definindo fundo em uma linha

Ao usar a propriedade `background` abreviada, a ordem dos valores da propriedade deve ser.

```
background: color image repeat attachment position;
```

Se o valor de uma propriedade de fundo individual estiver ausente ou não for especificado ao usar a notação abreviada, o valor padrão para essa propriedade será usado.

Fontes CSS

Fontes de estilo com CSS

Escolher a fonte e o estilo corretos é crucial para a legibilidade do texto em uma página.

CSS fornece várias propriedades para definir o estilo da fonte do texto, incluindo alterar sua face, controlar seu tamanho e negrito, gerenciar variantes e assim por diante.

As propriedades de fonte são: `font-family`, `font-style`, `font-weight`, `font-size`, e `font-variant`.

Família de fontes – (family)

A propriedade `font-family` é usada para especificar a fonte a ser usada para renderizar o texto.

Esta propriedade pode conter vários nomes de fontes separados por vírgulas como um sistema de *fallback*, de forma que se a primeira fonte não estiver disponível no sistema do usuário, o navegador tenta usar a segunda e assim por diante.

Portanto, liste a fonte desejada primeiro e, a seguir, quaisquer fontes que possam preencher a primeira, se não estiver disponível. Você deve terminar a lista com uma família de fontes genérica que são cinco - `serif`, `sans-serif`, `monospace`, `cursive` e `fantasy`. Uma declaração típica de família de fontes pode ter a seguinte aparência:

```
body {
    font-family: Arial, Helvetica, sans-serif;
}
```

Figura 70 - CSS: Declarando fonte

As famílias de fontes mais comuns usadas em web design são *serif* e *sans-serif*, porque são mais adequadas para leitura. Enquanto as fontes *monoespaçadas* são comumente usadas para exibir código, porque neste tipo de letra cada letra ocupa o mesmo espaço que se parece com um texto datilografado.

As fontes *cursivas* parecem escrita cursiva ou manuscrita. A fonte *fantasy* representa a fonte artística, mas não é amplamente utilizada devido à baixa disponibilidade dos sistemas operacionais.

Diferença entre fontes Serif e Sans-serif

As fontes com serifa têm linhas ou traços pequenos nas extremidades dos caracteres, enquanto as fontes sem serifa são mais retas e não têm esses traços pequenos. Veja a ilustração a seguir.

Figura 71 - CSS: Fontes com e sem serif

Estilo de fonte – (style)

A propriedade `font-style` é usada para definir o estilo da face da fonte para o conteúdo de texto de um elemento.

O estilo da fonte pode ser `normal`, `italic` ou `oblique`. O valor padrão é `normal`.

Vamos experimentar o seguinte exemplo para entender como funciona basicamente:

```
1  p.normal {
2    font-style: normal;
3  }
4  p.italic {
5    font-style: italic;
6  }
7  p.oblique {
8    font-style: oblique;
9  }
```

Figura 72 - CSS: Aplicando estilo a uma fonte

Tamanho da fonte – (size)

A propriedade `font-size` é usada para definir o tamanho da fonte para o conteúdo do texto de um elemento.

Existem várias maneiras de especificar os valores do tamanho da fonte, por exemplo, com palavras-chave, porcentagem, pixels, ems, etc.

Configurando o tamanho da fonte com pixels

Definir o tamanho da fonte em valores de pixel (por exemplo, 14px, 16px, etc.) é uma boa escolha quando você precisa de precisão de pixel. Pixel é uma unidade absoluta de medida que especifica um comprimento fixo.

Definir o tamanho da fonte em pixel não é considerado muito acessível, pois o usuário não pode alterar o tamanho da fonte nas configurações do navegador. Por exemplo, usuários com

visão limitada ou baixa visão podem desejar definir o tamanho da fonte muito maior do que o tamanho especificado por você.

Portanto, você deve evitar usar os valores de pixels e, em vez disso, usar os valores relativos ao tamanho da fonte padrão do usuário, se quiser criar um design inclusivo.

Configurando o tamanho da fonte com EM

A unidade `em` se refere ao tamanho da fonte do elemento pai. Ao definir a propriedade `font-size` como `1em` então este será o tamanho da fonte que se aplica ao *pai do elemento* .

Portanto, se você definir `font-size` 20px no elemento body, `1em = 20px` e `2em = 40px`.

No entanto, se você não definiu o tamanho da fonte em nenhum lugar da página, então `font-size` é o padrão do navegador, que normalmente é 16px. Portanto, por padrão `1em = 16px`, e `2em = 32px`.

Vamos dar uma olhada no exemplo a seguir para entender como funciona basicamente:

```css
h1 {
    font-size: 2em;     /* 32px/16px=2em */
}
p {
    font-size: 0.875em;     /* 14px/16px=0.875em */
}
```

Figura 73 - CSS: Definindo tamanho de fonte com Unidade Relativa "em"

Usando a combinação de porcentagem e EM

Como você observou no exemplo acima, o cálculo dos valores usando `em` não parece simples. Para simplificar isso, uma técnica popular é definir o `font-size` para o elemento do corpo como `62.5%`(ou seja, 62,5% do padrão 16px), o que equivale a 10px, ou 0,625em.

Agora você pode definir o `font-size` para quaisquer elementos usando unidades `em`, com uma conversão fácil de lembrar, dividindo o valor de `px` por 10.
Desta forma `10px = 1em`, `12px = 1.2em`, `14px = 1.4em`, `16px = 1.6em`, e assim por diante. Vamos dar uma olhada no seguinte exemplo:

```css
body {
    font-size: 62.5%;    /* font-size 1em = 10px */
}
p {
    font-size: 1.4em;    /* 1.4em = 14px */
}
p span {
    font-size: 2em;    /* 2em = 28px */
}
```

Figura 74 - CSS: Aplicando % para definir tamanho do texto com "em"

Configurando o tamanho da fonte com Root EM

Para tornar as coisas ainda mais simples, o CSS3 introduziu a unidade `rem` (abreviação de "root em") que é sempre relativa ao tamanho da fonte do elemento raiz (`html`), independentemente de onde o elemento está no documento (ao contrário de `em` que é relativo ao pai tamanho da fonte do elemento).

Isso significa que `1rem` é equivalente ao tamanho da fonte do `html`, que é o `16px` por padrão na maioria dos navegadores.

```css
html {
    font-size: 62.5%;    /* font-size 1em = 10px */
}
p {
    font-size: 1.4rem;    /* 1.4rem = 14px */
}
p span {
    font-size: 2rem;    /* 2rem = 20px (not 28px) */
}
```

Figura 75 - CSS: Definindo fonte via "rem"

Configurando o tamanho da fonte com palavras-chave

CSS fornece várias palavras-chave que você pode usar para definir tamanhos de fonte.

Um tamanho de fonte absoluto pode ser especificado usando um dos seguintes palavras-chave: `xx-small`, `x-small`, `small`, `medium`, `large`, `x-large`, `xx-large`. Enquanto isso, um tamanho de fonte relativo pode ser especificado usando as palavras-chave: `smaller` ou `larger`.

Configurando o tamanho da fonte com unidades de viewport

Os tamanhos das fontes podem ser especificados usando unidades de viewport como `vw` ou `vh`.

As unidades da janela de visualização referem-se a uma porcentagem das dimensões da janela de visualização do navegador, onde 1vw = 1% da largura da janela de visualização e 1vh = 1% da altura da janela de visualização. Portanto, se a janela de visualização tiver 1600 pixels de largura, 1vw será de 16 pixels.

No entanto, há um problema com as unidades da janela de visualização. Em telas pequenas, as fontes tornam-se tão pequenas que dificilmente são legíveis. Para evitar isso, você pode utilizar a função CSS `calc()`, como esta:

```css
html {
    font-size: calc(1em + 1vw);
}
h1 {
    font-size: 3rem;
}
```

Figura 76 - CSS: Corrigindo tamanho de fonte de acordo com a resolução do dispositivo (viewport)

Neste exemplo, mesmo que a largura da janela de visualização seja 0, o tamanho da fonte seria de pelo menos 1em ou 16px.

Espessura da fonte – (weight)

A propriedade `font-weight` especifica o peso ou negrito da fonte.

Esta propriedade pode ter um dos seguintes valores: `normal`, `bold`, `bolder`, `lighter`, `100`, `200`, `300`, `400`, `500`, `600`, `700`, `800`, `900` e `inherit`.

Os valores numéricos de `100`- `900` especifica os pesos da fonte, onde cada número representa um peso maior do que seu predecessor. `400` é o mesmo que `normal` e `700` é o mesmo que `bold`.

Os valores `bolder` e `lighter` são relativos ao peso da fonte herdado, enquanto os outros valores, como `normal` e `bold` são pesos absolutos da fonte.

Variante de fonte – (variant)

A propriedade `font-variant` permite que o texto seja exibido em uma variação especial de `small-caps`.

`Small-caps` ou *letras maiúsculas pequenas* são ligeiramente diferentes das letras maiúsculas normais, em que as letras minúsculas aparecem como versões menores das letras maiúsculas correspondentes.

```
1  p {
2      font-variant: small-caps;
3  }
```

Figura 77 - CSS: Aplicando small-caps na variação da fonte

Texto CSS

Formatando Texto com CSS

CSS fornece várias propriedades que permitem definir vários estilos de texto, como cor, alinhamento, espaçamento, decoração, transformação, etc. de forma fácil e eficaz.

As propriedades de texto comumente usados são: `text-align`, `text-decoration`, `text-transform`, `text-indent`, `line-height`, `letter-spacing`, `word-spacing`, e muito mais. Essas propriedades fornecem controle preciso sobre a aparência visual dos *caracteres*, *palavras*, *espaços* e assim por diante.

Vamos ver como definir essas propriedades de texto para um elemento com mais detalhes.

Cor do texto

A cor do texto é definida pela propriedade `color`.

Alinhamento de Texto

A propriedade `text-align` é usada para definir o alinhamento horizontal do texto.

O texto pode ser alinhado de quatro maneiras: à esquerda, à direita, ao centro ou justificado (margens direita e esquerda).

```
1  h1 {
2      text-align: center;
3  }
4  p {
5      text-align: justify;
6  }
```

Figura 78 - CSS: Alinhamento de texto

Vamos dar uma olhada na ilustração a seguir para entender o que esses valores realmente significam.

Figura 79 - CSS: Alinhamento de texto - Exemplo geral

Decoração de Texto

A propriedade `text-decoration` é usada para definir ou remover decorações do texto.

Esta propriedade normalmente aceita um dos seguintes valores: `underline`, `overline`, `line-through`, e `none`. Você deve evitar sublinhar texto que não seja um link, pois pode confundir o visitante.

Removendo o sublinhado padrão dos links

A propriedade `text-decoration` é amplamente usada para remover o sublinhado padrão dos hiperlinks HTML. Você pode ainda fornecer algumas outras dicas visuais para destacá-lo do texto normal, por exemplo, usando borda pontilhada em vez de sublinhado sólido.

```
1  a {
2      text-decoration: none;
3      border-bottom: 1px dotted;
4  }
```

Figura 80 - CSS: Alterando o comportamento de um link com text-decoration

Transformação de Texto

A propriedade `text-transform` é usada para definir os casos de um texto.

O texto costuma ser escrito em maiúsculas e minúsculas. No entanto, em certas situações, você pode querer exibir seu texto em letras totalmente diferentes. Usando esta propriedade, você pode alterar o conteúdo do texto de um elemento para letras maiúsculas ou minúsculas, ou colocar a primeira letra de cada palavra em maiúscula sem modificar o texto original.

Recuo de Texto

A propriedade `text-indent` é usada para definir o recuo da primeira linha de texto em um bloco de texto. Normalmente, isso é feito inserindo o espaço vazio antes da primeira linha do texto.

O tamanho do recuo pode ser especificado usando porcentagem (%), valores de comprimento em pixels, ems, etc.

Espaçamento entre Letras

A propriedade `letter-spacing` é usada para definir espaçamento extra entre os caracteres do texto.

Esta propriedade pode assumir um valor de comprimento em pixels, ems, etc. Ela também pode aceitar valores negativos. Ao definir o espaçamento entre letras, um valor de comprimento indica o espaçamento além do espaço entre caracteres padrão.

Espaçamento entre Palavras

A propriedade `word-spacing` é usada para especificar espaçamento adicional entre as palavras.

Esta propriedade pode aceitar um valor de comprimento em pixels, ems, etc. Valores negativos também são permitidos.

Altura da linha

A propriedade `line-height` é usada para definir a altura da linha do texto.

Também é chamado de *entrelinha* e normalmente é usado para definir a distância entre as linhas de texto.

O valor desta propriedade pode ser um número, uma porcentagem (%) ou um comprimento em pixels, ems, etc.

Quando o valor é um número, a altura da linha é calculada multiplicando o tamanho da fonte do elemento pelo número. Enquanto, os valores percentuais são relativos ao tamanho da fonte do elemento.

Se o valor da propriedade `line-height` for maior que o valor de `font-size` para um elemento, essa diferença (chamada de "*entrelinha*") é cortada pela metade (chamada de "*meio-guia*") e distribuída uniformemente na parte superior e inferior da box.

```
1  p {
2      font-size: 14px;
3      line-height: 20px;
4      background-color: #f0e68c;
5  }
```

Figura 81 - CSS: Definindo altura da linha

Links CSS

Links de estilo com CSS

Links ou hiperlinks são uma parte essencial de um site. Ele permite que os visitantes naveguem pelo site. Portanto, definir o estilo adequado dos links é um aspecto importante da construção de um site amigável.

A ligação tem quatro estados diferentes - `link`, `visited`, `active` e `hover`. Esses quatro estados de um link podem ter estilos diferentes por meio do uso dos seletores de pseudoclasse de âncora a seguir.

- **a: link** - define estilos para links normais ou não visitados.
- **a: visitado** - define estilos para links que o usuário já visitou.
- **a: hover** - define estilos para um link quando o usuário posiciona o ponteiro do mouse sobre ele.
- **a: ativo** - define estilos para links quando eles estão sendo clicados.

Você pode especificar qualquer propriedade CSS que você gostaria por exemplo **color**, **font**, **background**, **border**, etc, para cada um desses seletores para customizar o estilo de links, assim como você faz com o texto normal.

```
a:link {    /* unvisited link */
    color: #ff0000;
    text-decoration: none;
    border-bottom: 1px solid;
}
a:visited {    /* visited link */
    color: #ff00ff;
}
a:hover {    /* mouse over link */
    color: #00ff00;
    border-bottom: none;
}
a:active {    /* active link */
    color: #00ffff;
}
```

Figura 82 - CSS: Estilizando links com CSS

A ordem na qual você está definindo o estilo para diferentes estados de links é importante, porque o que define por último tem precedência sobre as regras de estilo definidas anteriormente.

Modificando Estilos de Link Padrão

Em todos os principais navegadores da web, como Chrome, Firefox, Safari, etc., os links nas páginas da web têm sublinhados e usam as cores de link padrão do navegador, se você não definir os estilos exclusivamente para eles.

Por padrão, os links de texto aparecerão da seguinte forma na maioria dos navegadores:

- Um link não visitado como texto azul sublinhado.
- Um link visitado como texto roxo sublinhado.
- Um link ativo como texto sublinhado em vermelho.

No entanto, não há alteração na aparência do link no caso do estado de foco. Permanece azul, roxo ou vermelho, dependendo do estado (ou seja, não visitado, visitado ou ativo) em que se encontram.

Agora vamos ver como personalizar os links substituindo seu estilo padrão.

Configurando a cor personalizada dos links

Basta usar a propriedade `color` para definir a cor de sua escolha para os diferentes estados de um link. Mas, ao escolher as cores, certifique-se de que o usuário possa diferenciar claramente entre texto normal e links.

```css
a:link {
    color: #1ebba3;
}
a:visited {
    color: #ff00f4;
}
a:hover {
    color: #a766ff;
}
a:active {
    color: #ff9800;
}
```

Figura 83 - CSS: Definindo cores de links

Removendo o sublinhado padrão dos links

Se você não gosta do sublinhado padrão nos links, pode simplesmente usar a propriedade `text-decoration` para se livrar dele. Alternativamente, você pode aplicar outros estilos aos links, como cor de fundo, borda inferior, fonte em negrito, etc. para torná-lo um pouco melhor do texto normal.

O exemplo a seguir mostra como remover ou definir sublinhado para diferentes estados de um link.

```css
a:link, a:visited {
    text-decoration: none;
}
a:hover, a:active {
    text-decoration: underline;
}
```

Figura 84 - CSS: Removendo decorations dos links

Fazendo os links de texto parecerem botões

Você também pode fazer com que seus links de texto comuns se pareçam com um botão usando CSS. Para fazer isso, precisamos de utilizar algumas propriedades tais como `background-color`, `border`, `display`, `padding`, etc.

Vamos dar uma olhada no exemplo a seguir e ver como ele realmente funciona:

```css
a:link, a:visited {
    color: white;
    background-color: #1ebba3;
    display: inline-block;
    padding: 10px 20px;
    border: 2px solid #099983;
    text-decoration: none;
    text-align: center;
    font: 14px Arial, sans-serif;
}
a:hover, a:active {
    background-color: #9c6ae1;
    border-color: #7443b6;
}
```

Figura 85 - CSS: Transformando links texto em botões (aparência)

Listas CSS

Tipos de listas HTML

Existem três tipos diferentes de lista em HTML:

- **Listas não ordenadas** - uma lista de itens, onde todos os itens da lista são marcados com marcadores.
- **Listas ordenadas** - uma lista de itens, onde cada item da lista é marcado com números.
- **Lista de definição** - uma lista de itens, com uma descrição de cada item.

Consulte o parágrafo sobre listas HTML para aprender mais sobre as listas e como criá-las.

Listas de estilo com CSS

CSS fornece várias propriedades para estilizar e formatar as listas não ordenadas e ordenadas mais comumente usadas. Essas propriedades de lista CSS geralmente permitem que você:

- Controle a forma ou aparência do marcador.
- Especifique uma imagem para o marcador em vez de um marcador ou número.
- Defina a distância entre um marcador e o texto na lista.

Especifique se o marcador aparecerá dentro ou fora da caixa que contém os itens da lista.

Na seção a seguir, discutiremos as propriedades que podem ser usadas para definir o estilo de listas HTML.

Alterar o tipo de marcador das listas

Por padrão, os itens em uma lista ordenada são numerados com algarismos arábicos (1, 2, 3, 5 e assim por diante), enquanto em uma lista não ordenada, os itens são marcados com marcadores redondos (•).

Mas, você pode alterar esse tipo de marcador de lista padrão para qualquer outro tipo, como algarismos romanos, letras latinas, círculo, quadrado e assim por diante, usando a propriedade `list-style-type`.

Alterar a posição dos marcadores de lista

Por padrão, os marcadores de cada item da lista são posicionados como `outside` em suas caixas de exibição.

No entanto, você também pode posicionar esses marcadores ou marcadores dentro das caixas de exibição do item da lista usando a propriedade `list-style-position` junto com o valor `inside`. Nesse caso, as linhas serão quebradas sob o marcador em vez de serem recuadas.

```
1  ol.in li {
2      list-style-position: inside;
3  }
4  ol.out li {
5      list-style-position: outside;
6  }
```

Figura 86 - CSS: Alterando posição de marcadores de listas

Vamos dar uma olhada na ilustração a seguir para entender como os marcadores ou marcadores são posicionados.

Figura 87 - CSS: Entendendo list-style-position

Usando imagens como marcadores de lista

Você também pode definir uma imagem como um marcador de lista usando a propriedade `list-style-image`.

A regra de estilo no exemplo a seguir atribui uma imagem PNG transparente "arrow.png" como o marcador de lista para todos os itens na lista não ordenada.

```
ul li {
    list-style-image: url("images/bullet.png");
}
```

Figura 88 - CSS: Usando imagens como marcadores de lista

Às vezes, a propriedade `list-style-image` pode não reproduzir o resultado esperado. Como solução alternativa, você pode definir marcadores de imagem por meio da propriedade `background-image`. Primeiro, defina a lista para não ter marcadores. Em seguida, defina uma imagem de fundo não repetitiva para o elemento da lista.

O exemplo a seguir exibe os marcadores de imagem igualmente em todos os navegadores:

```
ul {
    list-style-type: none;
}
ul li {
    background-image: url("images/bullet.png");
    background-position: 0px 5px;    /* X-pos Y-pos (from top-left) */
    background-repeat: no-repeat;
    padding-left: 20px;
}
```

Figura 89 - CSS: Alternativa ao list-style-image para navegadores que não seguem a convenção W3C

Configurando todas as propriedades da lista de uma vez

A propriedade `list-style` é um atalho para definir todas as três propriedades `list-style-type`, `list-style-image` e `list-style-position` de uma lista em um lugar.

A regra de estilo a seguir define todas as propriedades da lista em uma única declaração.

```css
ul {
    list-style: square inside url("images/bullet.png");
}
```

Figura 90 - CSS: List-style no método inline-shorthand

Criação de menus de navegação usando listas

Listas HTML são frequentemente usadas para criar uma barra de navegação horizontal ou menu que normalmente aparece na parte superior de um site. Mas, como os itens da lista são elementos de bloco, para exibi-los in-line, precisamos usar a propriedade `display`. Vamos experimentar um exemplo para ver como realmente funciona:

```css
ul {
    padding: 0;
    list-style: none;
    background: #f2f2f2;
}
ul li {
    display: inline-block;
}
ul li a {
    display: block;
    padding: 10px 25px;
    color: #333;
    text-decoration: none;
}
ul li a:hover {
    color: #fff;
    background: #939393;
}
```

Figura 91 - CSS: Criando um menu de navegação usando listas

Tabelas CSS

Tabelas de estilo com CSS

As tabelas são normalmente usadas para exibir dados tabulares, como relatórios financeiros.

Mas quando você cria uma tabela HTML sem estilos ou atributos, os navegadores as exibem sem nenhuma borda. Com CSS você pode melhorar muito a aparência de suas tabelas.

CSS fornece várias propriedades que permitem controlar o layout e a apresentação dos elementos da tabela.

Adicionando Bordas a Tabelas

A propriedade `border` é a melhor forma de definir as bordas das tabelas.

O exemplo seguinte irá definir uma borda preta para os elementos `<table>`, `<th>` e `<td>`.

```css
table, th, td {
    border: 1px solid black;
}
```

Figura 92 - CSS: Exemplo de estilização de tabela

Por padrão, o navegador desenha uma borda ao redor da tabela, bem como ao redor de todas as células, com algum espaço entre eles, o que resulta em uma borda dupla. Para se livrar desse problema de borda dupla, você pode simplesmente recolher as bordas das células da tabela adjacentes e criar bordas de linha única limpas.

Vamos dar uma olhada na ilustração a seguir para entender como uma borda é aplicada a uma mesa.

ID	Name	Age
1	John Carter	30
2	Harry Potter	11
3	Peter Parker	21

Separate Border (Default)

ID	Name	Age
1	John Carter	30
2	Harry Potter	11
3	Peter Parker	21

Collapse Border

Figura 93 - CSS: Formatação da tabela padrão e Tabela com CSS aplicado

Recolhendo bordas da tabela

Existem dois modelos distintos para definir bordas nas células da tabela em CSS: *separate* and *collapse*.

No modelo de borda *separate*, que é o padrão, cada célula da tabela tem suas próprias bordas distintas, enquanto no modelo de borda *collapse*, as células adjacentes da tabela compartilham uma borda comum. Você pode definir o modelo de borda para uma tabela HTML usando a propriedade `border-collapse`.

As seguintes regras de estilo reduzirão as bordas das células da tabela e aplicarão uma borda preta de pixel.

```css
table {
    border-collapse: collapse;
}
th, td {
    border: 1px solid black;
}
```

Figura 94 - CSS: Alterando comportamento da borda da tabela

Ajustando o Espaço dentro das Tabelas

Por padrão, o navegador cria as células da tabela grandes o suficiente para conter os dados nas células.

Para adicionar mais espaço entre o conteúdo da célula da tabela e as bordas da célula, você pode simplesmente usar a propriedade `padding`.

Você também pode ajustar o espaçamento entre as bordas das células usando a propriedade `border-spacing`, se as bordas de sua tabela estiverem separadas (que é o padrão).

As seguintes regras de estilo aplicam o espaçamento de 10 pixels entre todas as bordas de uma tabela:

```
1  table {
2      border-spacing: 10px;
3  }
```

Figura 95 - CSS: Ajustando espaçamento da tabela

Definir a largura e altura da mesa

Por padrão, uma tabela irá renderizar apenas larga e alta o suficiente para conter todo o seu conteúdo.

No entanto, você também pode definir a largura e a altura da tabela, bem como suas células, usando explicitamente a propriedade `width` e `height`.

Controlando o Layout da Tabela

Uma tabela se expande e se contrai para acomodar os dados contidos nela. Este é o comportamento padrão. Conforme os dados são preenchidos na tabela, eles continuam a se expandir enquanto houver espaço. Às vezes, no entanto, é necessário definir uma largura fixa para a mesa para gerenciar o layout.

Você pode fazer isso com a ajuda da propriedade `table-layout`. Esta propriedade define o algoritmo a ser usado para o layout das células, linhas e colunas da tabela. Esta propriedade assume um de dois valores:

- **auto** - usa um algoritmo automático de layout de tabela. Com este algoritmo, as larguras da tabela e suas células são ajustadas para caber no conteúdo. Este é o valor padrão.
- **fixed** - Usa o algoritmo de layout de tabela fixa. Com este algoritmo, o layout horizontal da tabela não depende do conteúdo das células; depende apenas da largura da tabela, da largura das colunas e das bordas ou espaçamento das células. Normalmente é mais rápido do que automático.

As regras de estilo no exemplo a seguir especificam que a tabela HTML é disposta usando o algoritmo de layout fixo e tem uma largura fixa de 300 pixels. Vamos experimentar e ver como funciona:

```
1  table {
2      width: 300px;
3      table-layout: fixed;
4  }
```

Figura 96 - CSS:Controlando o layout da tabela

Alinhando o texto dentro das células da tabela

Você pode alinhar o conteúdo do texto dentro das células da tabela horizontal ou verticalmente.

Alinhamento horizontal do conteúdo da célula

Para o alinhamento horizontal do texto dentro das células da tabela, você pode usar a propriedade `text-align` da mesma forma que usa com outros elementos. Você alinha o texto à esquerda, à direita, ao centro ou justificado.

As seguintes regras de estilo alinharão à esquerda o texto dentro dos elementos `<th>`.

```
th {
    text-align: left;
}
```

Figura 97 - CSS: Alinhamento horizontal do conteúdo da célula

Alinhamento vertical do conteúdo da célula

Da mesma forma, você pode alinhar verticalmente o conteúdo dentro dos elementos `<th>` e `<td>` na parte superior, inferior ou central usando a propriedade `vertical-align`. O alinhamento vertical padrão é o meio.

As regras de estilo a seguir alinharão verticalmente o texto dentro dos elementos `<th>` por baixo.

```
th {
    height: 40px;
    vertical-align: bottom;
}
```

Figura 98 - CSS: Alinhamento vertical do conteúdo da célula

Controlando a posição da legenda da tabela

Você pode definir a posição vertical de uma legenda de tabela usando a propriedade `caption-side`.

A legenda pode ser colocada na parte superior ou inferior da tabela. A posição padrão é superior.

Manipulação de células vazias

Em tabelas que usam modelo de borda separada, que é o padrão, você também pode controlar a renderização das células que não têm conteúdo visível usando a propriedade `empty-cell`.

Esta propriedade aceita um valor de `show` ou `hide`. O valor padrão é `show`, que renderiza células vazias como células normais, mas se o valor `hide` for especificado, nenhuma borda ou plano de fundo será desenhado ao redor das células vazias. Vamos tentar um exemplo para entender como realmente funciona:

```
table {
    border-collapse: separate;
    empty-cells: hide;
}
```

Figura 99 - CSS: Controlando a exibição de celulas vazias

Criação de tabelas listradas (zebradas)

Definir cores de fundo diferentes para linhas alternadas é uma técnica popular para melhorar a legibilidade de tabelas que possuem grande quantidade de dados. Isso é comumente conhecido como zebra-striping a table.

Você pode simplesmente obter esse efeito usando o seletor de pseudo-classe `:nth-child()`.

As seguintes regras de estilo destacarão todas as linhas ímpares no corpo da tabela.

```
tbody tr:nth-child(odd) {
    background-color: #f2f2f2;
}
```

Figura 100 - CSS: Aplicação de pseudo-classe para criar tabela zebrada

Uma tabela listrada de zebra normalmente se parece com a imagem a seguir.

Row	First Name	Last Name	Email
1	Clark	Kent	clarkkent@mail.com
2	John	Carter	johncarter@mail.com
3	Peter	Parker	peterparker@mail.com

Figura 101 - CSS: Exemplo de tabela zebrada

Tornando uma Tabela Responsiva

As tabelas não são responsivas por natureza. No entanto, para oferecer suporte a dispositivos móveis, você pode adicionar capacidade de responsividade às suas tabelas, permitindo a rolagem horizontal em telas pequenas. Para fazer isso, basta envolver sua tabela com um elemento `<div>` e aplicar o estilo `overflow-x: auto;` conforme mostrado abaixo:

```
<div style="overflow-x: auto;">
    <table>
        ... table content ...
    </table>
</div>
```

Figura 102 - CSS: Tornando uma tabela responsiva

CSS Box Model

O que é Box Model?

Cada elemento que pode ser exibido em uma página da web é composto por uma ou mais caixas retangulares. O modelo de caixa CSS geralmente descreve como essas caixas retangulares são dispostas em uma página da web. Essas caixas podem ter propriedades diferentes e interagir umas com as outras de maneiras diferentes, mas cada caixa tem uma ***área de conteúdo*** (***content area***) e áreas opcionais de ***preenchimento (padding)***, ***borda (border)*** e ***margem (margin-areas)*** ao redor.

O diagrama a seguir demonstra como as propriedades CSS de largura, altura, preenchimento, borda e margem determinam quanto espaço um elemento pode ocupar em uma página da web.

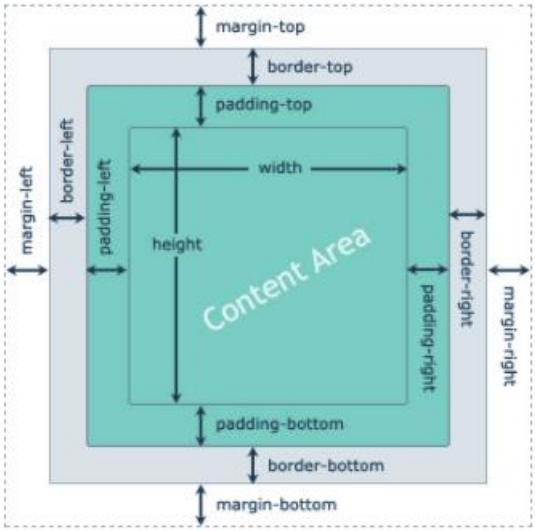

Figura 103 - CSS: Box Model

Preenchimento (padding) é o espaço transparente entre o conteúdo do elemento e sua borda (ou borda da caixa, se não houver borda), enquanto a margem (margin) é o espaço transparente ao redor da borda.

Além disso, se um elemento tiver a cor de fundo, ele será visível por meio de sua área de preenchimento (content). A área da margem (margin-area) sempre permanece transparente, não é afetada pela cor de fundo do elemento, no entanto, faz com que a cor de fundo do elemento pai seja vista através dela.

Largura e altura dos elementos

Normalmente, quando você define a largura e a altura de um elemento usando propriedades CSS `width` e `height`, na realidade, você está apenas definindo a largura e a altura da área de conteúdo desse elemento. A largura e altura reais da caixa do elemento dependem de vários fatores.

O espaço real que a caixa de um elemento pode ocupar em uma página da web é calculado assim:

Tamanho da Caixa	**Propriedades CSS**
Largura total	`width + padding-left + padding-right + border-left + border-right + margin-left + margin-right`
Altura total	`height + padding-top + padding-bottom + border-top + border-bottom + margin-top + margin-bottom`

Figura 104 - CSS: Largura e Altura dos elementos

Agora, vamos experimentar o seguinte exemplo para entender como o modelo de caixa realmente funciona:

```css
div {
    width: 300px;
    height: 200px;
    padding: 15px; /* set padding for all four sides */
    border: 10px solid black; /* set border for all four sides */
    margin: 20px auto; /* set top and bottom margin to 20 pixels, and left and right margin to auto */
}
```

Figura 105 - CSS: Exemplo de CSS Box Model

Dimensão CSS

Definindo as dimensões do elemento

CSS tem várias propriedades de dimensão, tais como `width`, `height`, `max-width`, `min-width`, `max-height`, e `min-height` que lhe permite controlar a largura e altura de um elemento.

Definir a largura e altura

A propriedade `width` e `height` definem a largura e a altura da área de conteúdo de um elemento.

Essa largura e altura não incluem preenchimentos, bordas ou margens.

As regras de estilo acima atribuem uma largura fixa de 300 pixels e altura de 200 pixels ao elemento `<div>`.

As propriedades `width` e `height` podem assumir os seguintes valores:

- *length* - especifica a largura em px, em, rem, pt, cm, etc.
- *%* - especifica uma largura em porcentagem (%) da largura do elemento que o contém.
- auto - o navegador calcula uma largura adequada para o elemento.
- initial - define a largura e a altura para seu valor padrão, que é `auto`.
- inherit - especifica que a largura deve ser herdada do elemento pai.

Você não pode especificar valores negativos para as propriedades de largura e altura.

Definir largura e altura máximas

Você pode usar a propriedade `max-width` e `max-height` para especificar a largura e altura máximas da área de conteúdo. Essa largura e altura máximas não incluem preenchimentos, bordas ou margens.

Um elemento não pode ser mais largo do que o valor de `max-width`, mesmo se o valor `width` da propriedade for definido como algo maior. Por exemplo, se `width` for definido como 300px e `max-width` como 200px, a largura real do elemento será 200px.

Da mesma forma, um elemento que teve `max-height` aplicado nunca será mais alto do que o valor especificado, mesmo se a propriedade `height` for definida como algo maior. Por

exemplo, se o `height` for definido como 200px e o `max-height` definido como 100px, a altura real do elemento será 100px.

Definir largura e altura mínimas

Você pode usar a propriedade `min-width` e `min-height` para especificar a largura e altura mínimas da área de conteúdo. Essa largura e altura mínimas não incluem preenchimentos, bordas ou margens.

Um elemento não pode ser mais estreito do que o valor de `min-width`, mesmo se o valor `width` da propriedade for definido como algo menor. Por exemplo, se `width` for definido como 300px e `min-width` como 400px, a largura real do elemento será 400px.

Da mesma forma, um elemento ao qual `min-height` é aplicado nunca será menor do que o valor especificado, mesmo se a propriedade `height` for definida como algo menor. Por exemplo, se `height` for definido como 200px e `min-height` como 300px, a altura real do elemento será 300px.

Definir um intervalo de largura e altura

As propriedades `min-width` e `min-height` são frequentemente usadas em combinação com as propriedades `max-width` e `max-height` para produzir um intervalo de largura e altura para um elemento.

Isso pode ser muito útil para criar um design flexível. No exemplo a seguir, a largura mínima do elemento `<div>` seria 300px e pode se estender horizontalmente até no máximo 500px.

```
1  div {
2      min-width: 300px;
3      max-width: 500px;
4  }
```

Figura 106 - CSS: Definindo intervalo mínimo e máximo da largura

Da mesma forma, você pode definir um intervalo de altura para um elemento. No exemplo abaixo, a altura mínima do elemento `<div>` seria 300px e pode se estender verticalmente até no máximo 500px.

```
1  div {
2      min-height: 300px;
3      max-height: 500px;
4  }
```

Figura 107 - CSS: Definindo intervalo mínimo e máximo da altura

Padding CSS

Propriedades de preenchimento CSS

As propriedades de preenchimento CSS permitem definir o espaçamento entre o conteúdo de um elemento e sua borda (ou a borda da caixa do elemento, se não houver borda definida).

O preenchimento é afetado pelo do elemento `background-color`. Por exemplo, se você definir a cor de fundo de um elemento, ele ficará visível na área de preenchimento.

Definir Paddings para Lados Individuais

Você pode especificar os guarnição para os lados individuais de um elemento como superior, direita, inferior e lados esquerdos, usando as propriedades `padding-top`, `padding-right`, `padding-bottom`, e `padding-left`, respectivamente.

As propriedades de preenchimento podem ser especificadas usando os seguintes valores:

- *length* - especifica um preenchimento em px, em, rem, pt, cm, etc.
- *%* - especifica um preenchimento em porcentagem (%) da largura do elemento que o contém.
- inherit - especifica que o preenchimento deve ser herdado do elemento pai.

Ao contrário da margem, os valores das propriedades de preenchimento não podem ser negativos.

A propriedade Padding Shorthand

A propriedade `padding` é uma propriedade abreviada para evitar a criação de preenchimento de cada um dos lados separadamente, ou seja, `padding-top`, `padding-right`, `padding-bottom` e `padding-left`.

Vamos dar uma olhada no exemplo a seguir para entender como funciona basicamente:

```
1  h1 {
2      padding: 50px; /* apply to all four sides */
3  }
4  p {
5      padding: 25px 75px; /* vertical | horizontal */
6  }
7  div {
8      padding: 25px 50px 75px; /* top | horizontal | bottom */
9  }
10 pre {
11     padding: 25px 50px 75px 100px; /* top | right | bottom | left */
12 }
```

Figura 108 - CSS: Definindo padding de forma inline / shorthand

Essa notação abreviada pode ter um, dois, três ou quatro valores separados por espaços em branco.

- Se ***um*** *valor* for especificado, ele será aplicado aos **quatro lados**.
- Se ***dois*** *valores* forem especificados, o primeiro valor será aplicado ao lado **superior e inferior**, e o segundo valor será aplicado ao lado **direito e esquerdo** da caixa do elemento.
- Se ***três*** *valores* forem especificados, o primeiro valor será aplicado na parte **superior**, o segundo valor será aplicado nos lados **direito e esquerdo** e o último valor será aplicado na **parte inferior**.
- Se ***quatro*** *valores* forem especificados, eles serão aplicados na parte **superior**, **direita**, **inferior** e **esquerda** da caixa do elemento, respectivamente, na ordem especificada.

Recomenda-se usar as propriedades abreviadas, pois isso o ajudará a economizar algum tempo, evitando a digitação extra e tornar seu código CSS mais fácil de seguir e manter.

Efeito de preenchimento e borda no layout

Ao criar layouts de página da web, adicionar um preenchimento ou borda aos elementos às vezes produz resultados inesperados, porque o preenchimento e a borda são adicionados à largura e à altura da caixa gerada pelo elemento, como você aprendeu no capítulo de CSS Box Model .

Por exemplo, se você definir a largura de um elemento `<div>` para 100% e também aplicar o preenchimento ou borda esquerda direita nele, a barra de rolagem horizontal aparecerá.

Para evitar que o preenchimento e a borda alterem a largura e a altura da caixa do elemento, você pode usar a propriedade `box-sizing`.

Borda CSS

Propriedades de borda CSS

As propriedades da borda CSS permitem definir a área da borda da caixa de um elemento.

As bordas aparecem diretamente entre a margem e o preenchimento de um elemento. A borda pode ser um estilo predefinido como linha sólida, linha pontilhada, linha dupla, etc. ou uma imagem.

A seção a seguir descreve como definir o estilo, a cor e a largura da borda.

Compreendendo os diferentes estilos de borda

A propriedade `border-style` define o estilo da borda de uma caixa, tais como: `solid`, `dotted`, etc. É um atalho para definir o estilo de linha para todos os quatro lados da fronteira elementos.

A propriedade `border-style` pode ter os seguintes valores: `none`, `hidden`, `solid`, `dashed`, `dotted`, `double`, `inset`, `outset`, `groove`, e `ridge`. Agora, vamos dar uma olhada na ilustração a seguir, ela dá uma ideia das diferenças entre os tipos de estilo de borda.

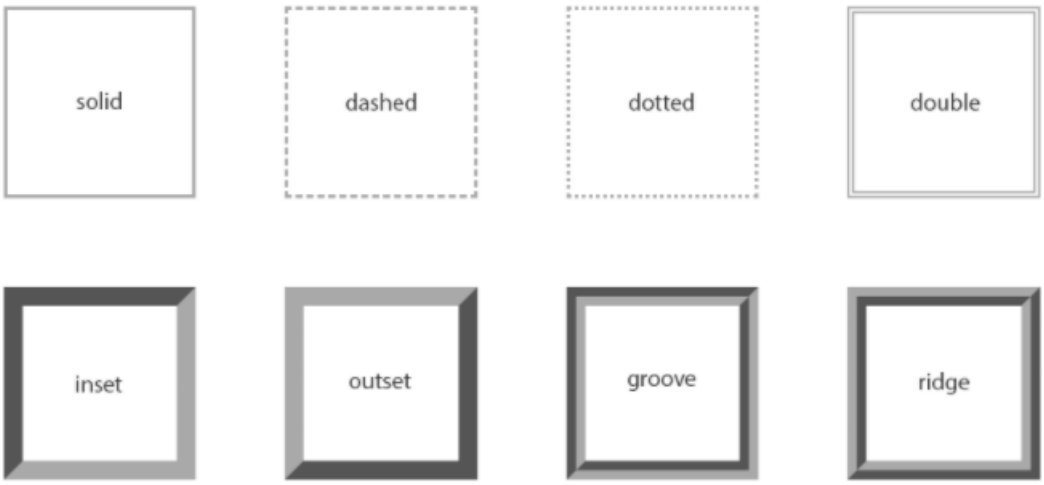

Figura 109 - CSS: Bordas em CSS

Os valores `none` e `hidden` não exibem bordas, no entanto, há uma ligeira diferença entre esses dois valores. No caso de célula de tabela e redução de borda, o valor `none` tem a prioridade *mais baixa* , enquanto o valor `hidden` tem a prioridade *mais alta* , se qualquer outra borda conflitante for definida.

Os valores `inset`, `outset`, `groove`, e `ridge` criam um efeito 3D como o que depende essencialmente do valor de `border-color`. Isso normalmente é obtido criando uma "sombra" a partir de duas cores que são ligeiramente mais claras e mais escuras do que a cor da borda.

Configurando a largura da borda

A propriedade `border-width` especifica a largura da área da borda. É uma propriedade abreviada para definir a espessura de todos os quatro lados da borda de um elemento ao mesmo tempo.

Especificando a cor da borda

A propriedade `border-color` especifica o `color` da área da borda. Esta também é uma propriedade abreviada para definir a cor de todos os quatro lados da borda de um elemento.

As regras de estilo a seguir adicionam uma borda sólida de cor vermelha ao redor dos parágrafos.

A propriedade da abreviação de borda

A propriedade `border` é um atalho para definir uma ou mais das propriedades de fronteira individuais `border-width`, `border-style`e `border-color` em uma única regra.

```
1  p {
2      border: 5px solid #00ff00;
3  }
```

Figura 110 - CSS: Aplicando borda de forma inline

Se o valor de uma propriedade de borda individual for omitido ou não especificado durante a configuração da propriedade abreviada de borda, o valor padrão dessa propriedade será usado, se houver.

Por exemplo, se o valor da propriedade `border-color` estiver faltando ou não for especificado ao definir a borda, a propriedade `color` do elemento será usada como o valor para a cor da borda.

Mas, no caso de `border-style`, omitir o valor fará com que nenhuma borda seja exibida, porque o valor padrão para esta propriedade é `none`.

Margem CSS

Propriedades de margem CSS

As propriedades de margem CSS permitem que você defina o espaçamento ao redor da borda da caixa de um elemento (ou a borda da caixa do elemento, se ela não tiver uma borda definida).

A margem de um elemento não é afetada por ela `background-color`, ela é sempre transparente. No entanto, se o elemento pai tiver a cor de fundo, ele será visível através de sua área de margem.

Definindo Margens para Lados Individuais

Você pode especificar as margens para os lados individuais de um elemento como superior, direita, inferior e lados esquerdos, usando o CSS `margin-top`, `margin-right`, `margin-bottom`, e as propriedades `margin-left`, respectivamente.

```css
h1 {
    margin-top: 50px;
    margin-bottom: 100px;
}
p {
    margin-left: 75px;
    margin-right: 75px;
}
```

Figura 111 - CSS: Definindo margens

As propriedades da margem podem ser especificadas usando os seguintes valores:

- *length* - especifica uma margem em px, em, rem, pt, cm, etc.
- *%* - especifica uma margem em porcentagem (%) da largura do elemento que o contém.
- auto - o navegador calcula uma margem adequada para usar.
- inherit - especifica que a margem deve ser herdada do elemento pai.

Você também pode especificar margens negativas em um elemento, por exemplo:

- `margin: -10px;`
- `margin: -5%;`.

A propriedade Margin Shorthand

A propriedade `margin` é uma propriedade abreviada para evitar a criação margem de cada lado separadamente, ou seja, `margin-top`, `margin-right`, `margin-bottom` e `margin-left`.

```css
h1 {
    margin: 50px; /* apply to all four sides */
}
p {
    margin: 25px 75px; /* vertical | horizontal */
}
div {
    margin: 25px 50px 75px; /* top | horizontal | bottom */
}
hr {
    margin: 25px 50px 75px 100px; /* top | right | bottom | left */
}
```

Figura 112 - CSS: Definindo margin de forma inline / shorthand

Essa notação abreviada pode ter um, dois, três ou quatro valores separados por espaços em branco.

- Se ***um*** *valor* for especificado, ele será aplicado aos **quatro lados**.
- Se ***dois*** *valores* forem especificados, o primeiro valor será aplicado ao lado **superior e inferior**, e o segundo valor será aplicado ao lado **direito e esquerdo** da caixa do elemento.
- Se ***três*** *valores* forem especificados, o primeiro valor será aplicado na parte **superior**, o segundo valor será aplicado nos lados **direito e esquerdo** e o último valor será aplicado na **parte inferior**.
- Se ***quatro*** *valores* forem especificados, eles serão aplicados na parte **superior**, **direita**, **inferior** e **esquerda** da caixa do elemento, respectivamente, na ordem especificada.

Recomenda-se usar as propriedades abreviadas, pois isso o ajudará a economizar algum tempo, evitando a digitação extra e tornar seu código CSS mais fácil de seguir e manter.

Centralização horizontal com margens automáticas

O valor `auto` da propriedade de margem informa ao navegador da web para calcular automaticamente a margem. Isso é comumente usado para centralizar um elemento horizontalmente dentro de um contêiner maior.

Vamos experimentar o seguinte exemplo para entender como funciona:

```
1  div {
2      width: 300px;
3      background: gray;
4      margin: 0 auto;
5  }
```

Figura 113 - CSS: Definindo margens automáticas

As regras de estilo acima permitem que o elemento `<div>` ocupe 300 pixels de todo o espaço horizontal disponível, e o espaço restante será dividido igualmente entre as margens esquerda e direita.

Tutorial de JavaScript

JavaScript é a linguagem de script do lado do cliente mais popular e amplamente usada. Os scripts do lado do cliente referem-se a scripts executados no navegador da web. JavaScript é projetado para adicionar interatividade e efeitos dinâmicos às páginas da web, manipulando o conteúdo retornado de um servidor da web.

JavaScript foi originalmente desenvolvido como LiveScript pela Netscape em meados de 1990. Posteriormente, foi renomeado para JavaScript em 1995 e tornou-se um padrão ECMA em 1997. Agora, o JavaScript é a linguagem de script do lado do cliente padrão para aplicativos baseados na web e é compatível com praticamente todos os navegadores disponíveis hoje, como o Google Chrome, Mozilla Firefox, Apple Safari, etc.

JavaScript é uma linguagem orientada a objetos e também possui algumas semelhanças na sintaxe com a linguagem de programação Java. Mas o JavaScript não está relacionado ao Java de forma alguma.

JavaScript é oficialmente mantido pela ECMA (European Computer Manufacturers Association) como ECMAScript. ECMAScript 6 (ou ES6) é a versão principal mais recente do padrão ECMAScript

O que você pode fazer com JavaScript

Existem muito mais coisas que você pode fazer com JavaScript.

- Você pode modificar o conteúdo de uma página da web adicionando ou removendo elementos.
- Você pode alterar o estilo e a posição dos elementos em uma página da web.
- Você pode monitorar eventos como clique do mouse, pairar, etc. e reagir a eles.
- Você pode executar e controlar transições e animações.
- Você pode criar pop-ups de alerta para exibir informações ou mensagens de aviso ao usuário.
- Você pode realizar operações com base nas entradas do usuário e exibir os resultados.
- Você pode validar as entradas do usuário antes de enviá-las ao servidor.

A lista não termina aqui, há muitas outras coisas interessantes que você pode fazer com JavaScript.

Introdução ao JavaScript

Aqui, você aprenderá como é fácil adicionar interatividade a uma página da web usando JavaScript. Mas, antes de começar, certifique-se de ter algum conhecimento prático de HTML e CSS.

Bem, vamos começar com a linguagem de script do lado do cliente mais popular.

Adicionando JavaScript às suas páginas da web

Normalmente, existem três maneiras de adicionar JavaScript a uma página da web:

- Incorporar o código JavaScript entre um par de tags `<script>` e `</script>`.

- Criar um arquivo JavaScript externo com a extensão `.js` e carregá-lo na página por meio do atributo `src` da `<script>`tag.
- Colocando o código JavaScript diretamente dentro de uma tag HTML usando a tag especial atributos como `onclick`, `onmouseover`, `onkeypress`, `onload`, etc.

Incorporando o código JavaScript

Você pode incorporar o código JavaScript diretamente em suas páginas da web, colocando-o entre as tags `<script>` e `</script>`. A tag `<script>` indica ao navegador que as instruções contidas devem ser interpretadas como script executável e não HTML. Aqui está um exemplo:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Incorporando JavaScript</title>
</head>
<body>
    <script>
    var saudacao = "Olá Mundo!";
    document.write(saudacao); // Imprime: Olá Mundo!
    </script>
</body>
</html>
```

Figura 114 - JavaScript: Incorporando JS a uma página web

O código JavaScript no exemplo acima simplesmente imprimirá uma mensagem de texto na página da web.

Chamando um arquivo JavaScript externo

Você também pode colocar seu código JavaScript em um arquivo separado com uma extensão `.js` e, em seguida, chamar esse arquivo em seu documento por meio do atributo `src` da tag `<script>`, como este:

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

Isso é útil se você deseja que os mesmos scripts estejam disponíveis para vários documentos. Isso evita que você repita a mesma tarefa indefinidamente e torna a manutenção do seu site muito mais fácil.

Bem, vamos criar um arquivo JavaScript chamado "hello.js" e colocar o seguinte código nele:

```
// Função para exibir uma mensagem
function digaOla() {
    alert("Olá Mundo!");
}

// Chamar função ao clicar no botão com id="myBtn"
document.getElementById("myBtn").onclick = digaOla;
```

Figura 115 - JavaScript: Chamando função num arquivo externo

Agora, você pode chamar esse arquivo JavaScript externo dentro de uma página da web usando a tag `<script>`, assim:

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Incluindo um arquivo JavaScript externo</title>
</head>
<body>
    <button type="button" id="myBtn">Clique aqui</button>
    <script src="/javascript/hello.js"></script>
</body>
</html>
```

Figura 116 - JavaScript: Chamando arquivo externo

Colocando o código JavaScript embutido

Você também pode colocar JavaScript código embutido, inserindo-o diretamente dentro da tag HTML usando a tag especial atributos como `onclick`, `onmouseover`, `onkeypress`, `onload`, etc.

No entanto, você deve evitar colocar uma grande quantidade de código JavaScript embutido, pois isso confunde seu HTML com JavaScript e torna seu código JavaScript difícil de manter.

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Executando um arquivo JavaScript inline</title>
</head>
<body>
    <button type="button" onclick="alert('Olá Mundo!')">Clique aqui</button>
</body>
</html>
```

Figura 117 - JavaScript: Executando javascript de forma inline

Posicionamento do script dentro do documento HTML

O elemento `<script>` pode ser colocado na seção `<head>` ou `<body>` de um documento HTML. Mas, idealmente, os scripts devem ser colocados no final da seção do corpo, antes da tag `</body>` de fechamento , isso fará com que suas páginas da web carreguem mais rápido, pois evita a obstrução da renderização inicial da página.

Cada tag `<script>` bloqueia o processo de renderização da página até que tenha totalmente baixado e executado o código JavaScript, portanto, colocá-los na seção principal (ou seja, no elemento `<head>`) do documento sem qualquer razão válida afetará significativamente o desempenho do seu site.

Diferença entre scripts do lado do cliente e do lado do servidor

Linguagens de script do lado do cliente, como JavaScript, VBScript, etc., são interpretadas e executadas pelo navegador da web, enquanto linguagens de script do lado do servidor, como PHP, ASP, Java, Python, Ruby, etc. são executadas no servidor da web e na saída enviado de volta para o navegador da web em formato HTML.

O script do lado do cliente tem muitas vantagens sobre a abordagem tradicional de script do lado do servidor. Por exemplo, você pode usar JavaScript para verificar se o usuário inseriu dados inválidos nos campos do formulário e mostrar notificações para erros de entrada de acordo em tempo real antes de enviar o formulário para o servidor da web para validação final de dados e processamento posterior, a fim de evitar usos desnecessários da largura de banda da rede e a exploração dos recursos do sistema do servidor.

Além disso, a resposta de um script do lado do servidor é mais lenta em comparação com um script do lado do cliente, porque os scripts do lado do servidor são processados no computador remoto e não no computador local do usuário.

Sintaxe JavaScript

Compreendendo a sintaxe do JavaScript

A sintaxe do JavaScript é o conjunto de regras que definem um programa JavaScript estruturado corretamente.

Um JavaScript consiste em instruções JavaScript que são colocadas nas tags `<script></script>` em uma página da web ou no arquivo JavaScript externo com extensão `.js`.

O exemplo a seguir mostra a aparência das instruções JavaScript:

```
1  var x = 5;
2  var y = 10;
3  var sum = x + y;
4  document.write(sum);
```

Figura 118 - JavaScript: Compreendendo a sintaxe do JS

Sensibilidade a maiúsculas e minúsculas em JavaScript

JavaScript diferencia maiúsculas de minúsculas. Isso significa que variáveis, palavras-chave de idioma, nomes de função e outros identificadores devem sempre ser digitados com letras maiúsculas consistentes.

Por exemplo, a variável `myVar` deve ser digitada `myVar` não `MyVar` ou `myvar`. Da mesma forma, o nome do método `getElementById()` deve ser digitado com o caso exato, não como `getElementByID()`.

```
1  var myVar = "Hello World!";
2  console.log(myVar);
3  console.log(MyVar);
4  console.log(myvar);
```

Figura 119 - JavaScript: Case Sensitive

Comentários de JavaScript

Um comentário é simplesmente uma linha de texto completamente ignorada pelo interpretador JavaScript. Os comentários são geralmente adicionados com o objetivo de fornecer informações extras relativas ao código-fonte. Isso não só o ajudará a entender seu código quando você cuidar de um período de tempo, mas também a outras pessoas que estão trabalhando com você no mesmo projeto.

O JavaScript suporta comentários de uma linha e também de várias linhas. Os comentários de uma linha começam com uma barra dupla (`//`), seguida pelo texto do comentário.

Enquanto um comentário de várias linhas começa com uma barra e um asterisco (`/*`) e termina com um asterisco e uma barra (`*/`).

Variáveis JavaScript

O que é variável?

Variáveis são fundamentais para todas as linguagens de programação. Variáveis são usadas para armazenar dados, como string de texto, números, etc. Os dados ou valores armazenados nas variáveis podem ser definidos, atualizados e recuperados sempre que necessário. Em geral, as variáveis são nomes simbólicos para valores.

Você pode criar uma variável com a `var` palavra - chave, enquanto o operador de atribuição (=) é usado para atribuir valor a uma variável, como este: `var varName = value;`

```
1  var name = "Peter Parker";
2  var age = 21;
3  var isMarried = false;
```

Figura 120 - JavaScript: Declarando variáveis

No exemplo acima, criamos três variáveis, a primeira atribuída a um valor de string, a segunda atribuída a um número, enquanto a última atribuída a um valor booleano. As variáveis podem conter diferentes tipos de dados, aprenderemos sobre elas no capítulo posterior.

Em JavaScript, as variáveis também podem ser declaradas sem ter nenhum valor inicial atribuído a elas. Isso é útil para variáveis que devem conter valores como entradas do usuário.

Declarando múltiplas variáveis de uma só vez

Além disso, você também pode declarar várias variáveis e definir seus valores iniciais em uma única instrução. Cada variável é separada por vírgulas, conforme demonstrado no exemplo a seguir:

```
// Declarando várias variaveis de uma vez
var name = "Peter Parker", age = 21, isMarried = false;

/*
  Declarações mais longas podem
  ser escritas para abranger
  várias linhas para melhorar a
  legibilidade
*/
var name = "Peter Parker",
age = 21,
isMarried = false;
```

Figura 121 - JavaScript: Declarando multiplas variáveis de uma vez só

As palavras-chave let e const (ES6)

O ES6 apresenta duas novas palavras-chaves `let` e `const` para declarar variáveis.

A palavra-chave `const` funciona exatamente da mesma forma que `let`, exceto que as variáveis declaradas usando a palavra-chave `const` não podem ser reatribuídas posteriormente no código.

```
// Declarando variaveis
let name = "Harry Potter";
let age = 11;
let isStudent = true;

// Declarando constante
const PI = 3.14;
console.log(PI); // 3.14

// Tentar reatribuir o valor
PI = 10; // vai gerar erro!!!
```

Figura 122 - JavaScript: Palavras-chave let e const

Ao contrário de `var`, que declara variáveis com escopo de função, `let` e as `const` declaram variáveis com escopo no nível de bloco (`{}`). Escopo de bloco significa que um novo escopo é criado entre um par de chaves `{}`.

Convenções de nomenclatura para variáveis JavaScript

Estas são as seguintes regras para nomear uma variável JavaScript:

- Um nome de variável deve começar com uma letra, sublinhado (`_`) ou cifrão (`$`).
- Um nome de variável não pode começar com um número.
- Um nome de variável pode conter apenas caracteres alfanuméricos (`A-z`, `0-9`) e sublinhados.
- Um nome de variável não pode conter espaços.
- Um nome de variável não pode ser uma palavra-chave de JavaScript ou uma palavra reservada de JavaScript.

Saída de geração de JavaScript

Gerando saída em JavaScript

Existem certas situações em que pode ser necessário gerar saída de seu código JavaScript. Por exemplo, você pode querer ver o valor da variável ou escrever uma mensagem no console do navegador para ajudá-lo a depurar um problema em seu código JavaScript em execução e assim por diante.

Em JavaScript, existem várias maneiras diferentes de gerar saída, incluindo escrever a saída na janela do navegador ou no console do navegador, exibir a saída em caixas de diálogo, gravar a saída em um elemento HTML, etc.

Gravando saída no console do navegador

Você pode facilmente enviar uma mensagem ou gravar dados no console do navegador usando o método `console.log()`. Este é um método simples, mas muito poderoso para gerar resultados detalhados.

Exibindo a saída em caixas de diálogo de alerta

Você também pode usar caixas de diálogo de alerta para exibir a mensagem ou dados de saída para o usuário. Uma caixa de diálogo de alerta é criada usando o método `alert()`

Gravando a saída na janela do navegador

Você pode usar o método `document.write()`para gravar o conteúdo no documento atual apenas enquanto esse documento está sendo analisado.

```
// Imprime um texto simples
  console.log("Olá Mundo!"); // Imprime: Olá Mundo!
// Exibe um texto simples
  alert("Olá Mundo!"); // Exibe: Olá Mundo!
// Escreve um texto simples
document.write("Olá Mundo!"); // Escreve: Olá Mundo!
```

Figura 123 - JavaScript: Exibe dados de formas diferentes com console.log, alert e document.write

Mas atenção! Se você usar o método `document.write()` depois que a página for carregada, ele substituirá todo o conteúdo existente nesse documento.

Inserindo saída dentro de um elemento HTML

Você também pode escrever ou inserir saída dentro de um elemento HTML usando a propriedade `innerHTML` do elemento . No entanto, antes de escrever a saída primeiro, precisamos selecionar o elemento usando um método como `getElementById()`, conforme demonstrado no exemplo a seguir:

```
<body>
  <p id="saudacao"></p>
  <p id="resultado"></p>

  <script>
  // Escrevendo texto dentro de um elemento
  document.getElementById("saudacao").innerHTML = "Olá Mundo!";

  // Exibindo o resultado de uma variavel em um elemento
  var x = 10;
  var y = 20;
  var sum = x + y;
  document.getElementById("resultado").innerHTML = sum;
  </script>

</body>
```

Figura 124 - JavaScript: Inserindo dados dentro de outros elementos

Tipos de dados JavaScript

Os tipos de dados basicamente especificam que tipo de dados pode ser armazenados e manipulados em um programa.

Existem seis tipos de dados básicos em JavaScript que podem ser divididos em três categorias principais: tipos de dados primitivos (ou *primários*), *compostos* (ou de *referência*) e *especiais* . String, Number e Boolean são tipos de dados primitivos. Object, Array e Function (que são todos tipos de objetos) são tipos de dados compostos. Considerando que Undefined e Null são tipos de dados especiais.

Os tipos de dados primitivos podem conter apenas um valor por vez, enquanto os tipos de dados compostos podem conter coleções de valores e entidades mais complexas.

O tipo de dados String

O tipo de dados *string* é usado para representar dados textuais (ou seja, sequências de caracteres). As strings são criadas usando aspas simples ou duplas em torno de um ou mais caracteres.

O tipo de dados do número

O tipo de dados de *número* é usado para representar números positivos ou negativos com ou sem casa decimal, ou números escritos usando notação exponencial, por exemplo, 1,5e-4 (equivalente a 1,5x10 $^{-4}$).

O tipo de dados número também inclui alguns valores especiais que são: `Infinity`, -`Infinity` e `NaN`. O infinito representa o infinito matemático ∞, que é maior do que qualquer número. O infinito é o resultado da divisão de um número diferente de zero por 0.

Enquanto `NaN` representa um valor especial "*não-um-número*" (Not A Number). É o resultado de uma operação matemática inválida ou indefinida, como tirar a raiz quadrada de -1 ou dividir 0 por 0, etc.

O tipo de dados booleano

O tipo de dados booleano pode conter apenas dois valores: `true` ou `false`. Normalmente é usado para armazenar valores como **sim** (`true`) ou **não** (`false`), **ativado** (`true`) ou desativado (`false`), etc.
Os valores booleanos também vêm como resultado de comparações em um programa.

```
1  var a = 2, b = 5, c = 10;
2
3  alert(b > a) // Output: true
4  alert(b > c) // Output: false
```

Figura 125 - JavaScript: Declarando boolenos

O tipo de dado indefinido

O tipo de dados indefinido só pode ter um valor - o valor especial `undefined`. Se uma variável foi declarada, mas não foi atribuída um valor, tem o valor `undefined`.

O tipo de dado nulo

Este é outro tipo de dados especial que pode ter apenas um valor, é o valor `null`. Um valor `null` significa que não há valor. Não é equivalente a uma string vazia (`""`) ou 0, simplesmente não é nada.
Uma variável pode ser explicitamente esvaziada de seu conteúdo atual atribuindo-lhe o valor `null`.

O tipo de dados do objeto

O `object` é um tipo de dados complexo que permite armazenar coleções de dados.
Um objeto contém propriedades, definidas como um par de valores-chave. Uma chave de propriedade (nome) é sempre uma string, mas o valor pode ser qualquer tipo de dados, como strings, números, booleanos ou tipos de dados complexos como matrizes, funções e outros objetos.

```
var emptyObject = {};
var person = {"name": "Clark", "lastname": "Kent", "age": "36"};

// Para uma melhor leitura
var car = {
    "modal": "BMW X3",
    "color": "white",
    "doors": 5
}
```

Figura 126 - JavaScript: Declarando um objeto

Você pode omitir as aspas ao redor do nome da propriedade se o nome for um nome JavaScript válido. Isso significa que as aspas são obrigatórias, `"first-name"` mas são opcionais em `firstname`. Portanto, o objeto *car* no exemplo acima também pode ser escrito como:

```
var car = {
    modal: "BMW X3",
    color: "white",
    doors: 5
}
```

Figura 127 - JavaScript: Outra forma de declarar um objeto

O tipo de dados Array

Uma matriz é um tipo de objeto usado para armazenar vários valores em uma única variável. Cada valor (também chamado de elemento) em uma matriz tem uma posição numérica, conhecida como seu índice, e pode conter dados de qualquer tipo de dados - **números**, **strings**, **booleanos**, **funções**, **objetos** e até mesmo **outras matrizes**. O índice da matriz começa em 0, de modo que o primeiro elemento da matriz é sempre `arr[0]` não é `arr[1]`.

A maneira mais simples de criar uma matriz é especificando os elementos da matriz como uma lista separada por vírgulas entre colchetes, conforme mostrado no exemplo abaixo:

```
var colors = ["Red", "Yellow", "Green", "Orange"];
var cities = ["London", "Paris", "New York"];

alert(colors[0]);   // Saída: Red
alert(cities[2]);   // Saída: New York
```

Figura 128 - JavaScript: Declarando um array

O tipo de dados Função

A função é um objeto "chamável" que executa um bloco de código. Como as funções são objetos, é possível atribuí-las a variáveis, conforme mostrado no exemplo abaixo:

```
var saudacao = function(){
    return "Olá Mundo!";
}

// Verifica o tipo de variavel que é "saudacao"
alert(typeof saudacao) // Saída: function
alert(saudacao());     // Saída: Olá Mundo!
```

Figura 129 - JavaScript: Tipo de Dados – Função

Na verdade, as funções podem ser usadas em qualquer lugar, qualquer outro valor pode ser usado. As funções podem ser armazenadas em variáveis, objetos e matrizes. Funções podem ser passadas como argumentos para outras funções e funções podem ser retornadas de funções.

O tipo de operador

O operador `typeof` pode ser usado para descobrir que tipo de dados uma variável ou operando contém. Pode ser usado com ou sem parênteses (`typeof(x)` ou `typeof x`).

O operador `typeof` é particularmente útil nas situações em que você precisa processar os valores de diferentes tipos de forma diferente, mas você precisa ter muito cuidado, pois pode produzir resultados inesperados em alguns casos, conforme demonstrado no exemplo a seguir:

```
// Numbers
typeof 15;  // Retorna: "number"
typeof 42.7;  // Retorna: "number"
typeof 2.5e-4;  // Retorna: "number"
typeof Infinity;  // Retorna: "number"
typeof NaN;  // Retorna: "number". Apesar de ser um "Not-A-Number"

// Strings
typeof '';  // Retorna: "string"
typeof 'hello';  // Retorna: "string"
typeof '12';  // Retorna: "string". Um número entre aspas é um tipo de string

// Booleans
typeof true;  // Retorna: "boolean"
typeof false;  // Retorna: "boolean"

// Undefined
typeof undefined;  // Retorna: "undefined"
typeof undeclaredVariable; // Retorna: "undefined"

// Null
typeof Null;  // Retorna: "object"

// Objects
typeof {name: "John", age: 18};  // Retorna: "object"

// Arrays
typeof [1, 2, 4];  // Retorna: "object"

// Functions
typeof function(){};  // Retorna: "function"
```

Figura 130 - JavaScript: Operador typeof

Como você pode ver claramente no exemplo acima, quando testamos o valor `null` usando o operador `typeof`, ele retornou "objeto" em vez de "nulo".

Este é um bug de longa data em JavaScript, mas como muitos códigos na web escritos em torno desse comportamento, e consertá-lo criaria muito mais problemas, a ideia de consertar esse problema foi rejeitada pelo comitê que projeta e mantém o JavaScript.

Operadores de JavaScript

O que são operadores em JavaScript

Operadores são símbolos ou palavras-chave que instruem o mecanismo JavaScript a realizar algum tipo de ação. Por exemplo, o + (símbolo de adição) é um operador que instrui o mecanismo JavaScript a adicionar duas variáveis ou valores, enquanto os símbolos igual a (==), maior que (>) ou menor que (<) são os operadores que instruem o mecanismo JavaScript a compare duas variáveis ou valores e assim por diante.

Operadores aritméticos JavaScript

Os operadores aritméticos são usados para realizar operações aritméticas comuns, como adição, subtração, multiplicação, etc. Aqui está uma lista completa dos operadores aritméticos do JavaScript:

Operador	Descrição	Exemplo	Resultado
+	Adição	x + y	Soma de x e y
-	Subtração	x - y	Diferença de x e y.
*	Multiplicação	x * y	Produto de x e y.
/	Divisão	x / y	Quociente de x e y
%	Módulo	x % y	Restante de x dividido por y

Figura 131 - JavaScript: Operadores aritméticos JavaScript

Operadores de atribuição de JavaScript

Os operadores de atribuição são usados para atribuir valores a variáveis.

Operador	Descrição	Exemplo	É o mesmo que
=	Atribuir	x = y	x = y
+=	Adicionar e atribuir	x += y	x = x + y
-=	Subtrair e atribuir	x -= y	x = x - y
*=	Multiplique e atribua	x *= y	x = x * y
/=	Divida e atribua quociente	x /= y	x = x / y
%=	Divida e atribua o módulo	x %= y	x = x % y

Figura 132 - JavaScript: Operadores de atribuição

Operadores de string JavaScript

Existem dois operadores que também podem ser usados para strings.

Operador	Descrição	Exemplo	Resultado
+	Concatenação	str1 + str2	Concatenação de str1 e str2
+=	Atribuição de concatenação	str1 += str2	Anexa str2 a str1

Figura 133 - JavaScript: Operadores de string

Operadores de incremento e decremento de JavaScript

Os operadores de incremento / decremento são usados para incrementar / decrementar o valor de uma variável.

Operador	Nome	Efeito
++x	Pré-incremento	Incrementa x por um e, em seguida, retorna x
x++	Pós-incremento	Retorna x, a seguir incrementa x por um
--x	Pré-decremento	Decrementa x por um e, em seguida, retorna x
x--	Pós-decremento	Retorna x, então diminui x em um

Figura 134 - JavaScript: Operadores de incremento e decremento

Operadores lógicos de JavaScript

Os operadores lógicos são normalmente usados para combinar declarações condicionais.

Operador	Nome	Exemplo	Resultado
&&	E	x && y	Verdadeiro se x e y forem verdadeiros
||	Ou	x || y	Verdadeiro se x ou y for verdadeiro
!	Não	!x	Verdadeiro se x não for verdadeiro

Figura 135 - JavaScript: Operadores Lógicos

Operadores de comparação de JavaScript

Os operadores de comparação são usados para comparar dois valores de maneira booleana.

Operador	Nome	Exemplo	Resultado
==	Igual	x == y	Verdadeiro se x for igual ay
===	Idêntico	x === y	Verdadeiro se x for igual ay e eles forem do mesmo tipo
!=	Não igual	x != y	Verdadeiro se x não for igual ay
!==	Não idênticos	x !== y	Verdadeiro se x não for igual ay ou se eles não forem do mesmo tipo
<	Menor que	x < y	Verdadeiro se x for menor que y
>	Maior que	x > y	Verdadeiro se x for maior que y
>=	Melhor que ou igual a	x >= y	Verdadeiro se x for maior ou igual a y
<=	Menos que ou igual a	x <= y	Verdadeiro se x for menor ou igual a y

Figura 136 - JavaScript: Operadores de comparação

Eventos JavaScript

Compreendendo eventos e manipuladores de eventos

Um evento é algo que acontece quando o usuário interage com a página da web, como quando ele clica em um link ou botão, insere texto em uma caixa de entrada ou área de texto, faz a seleção em uma caixa de seleção, pressiona uma tecla no teclado, move o ponteiro do

mouse , envia um formulário, etc. Em alguns casos, o próprio navegador pode acionar os eventos, como os eventos de carregamento e descarregamento da página.

Quando ocorre um evento, você pode usar um manipulador de eventos JavaScript (ou um ouvinte de eventos) para detectá-los e executar tarefas específicas ou conjunto de tarefas. Por convenção, os nomes dos manipuladores de eventos sempre começam com a palavra "on", então um manipulador de eventos para o evento click é chamado `onclick`, da mesma forma um manipulador de eventos para o evento load é chamado `onload`, um manipulador de eventos para o evento blur é chamado `onblur` e assim por diante.

Existem várias maneiras de atribuir um manipulador de eventos. A maneira mais simples é adicioná-los diretamente à marca de início dos elementos HTML usando os atributos especiais do manipulador de eventos. Por exemplo, para atribuir um manipulador de clique para um elemento de botão, podemos usar o atributo `onclick`, como este:

```
<button type="button" onclick="alert('Olá Mundo!')">Clique aqui</button>
```

Figura 137 - JavaScript: Compreendendo eventos e manipuladores de eventos

Em geral, os eventos podem ser categorizados em quatro grupos principais - **eventos de mouse** , **eventos de teclado** , eventos de formulário e **eventos de documento / janela**.

Eventos de mouse

Um evento de mouse é disparado quando o usuário clica em algum elemento, move o ponteiro do mouse sobre um elemento, etc. Aqui estão alguns eventos de mouse mais importantes e seu manipulador de eventos.

O evento Click (onclick)

O evento click ocorre quando um usuário clica em um elemento em uma página da web. Frequentemente, são elementos de formulário e links. Você pode manipular um evento de clique com um manipulador `onclick` de eventos.

O evento Contextmenu (oncontextmenu)

O evento contextmenu ocorre quando um usuário clica com o botão direito do mouse em um elemento para abrir um menu de contexto. Você pode manipular um evento contextmenu com um manipulador `oncontextmenu` de eventos.

O evento Mouseover (onmouseover)

O evento mouseover ocorre quando um usuário move o ponteiro do mouse sobre um elemento.

Você pode manipular o evento mouseover com o manipulador `onmouseover` de eventos.

O evento Mouseout (onmouseout)

O evento mouseout ocorre quando um usuário move o ponteiro do mouse para fora de um elemento.

Você pode manipular o evento mouseout com o manipulador `onmouseout` de eventos.

Eventos de teclado

Um evento de teclado é disparado quando o usuário pressiona ou solta uma tecla no teclado. Aqui estão alguns eventos de teclado mais importantes e seu manipulador de eventos.

O evento Keydown (onkeydown)

O evento keydown ocorre quando o usuário pressiona uma tecla no teclado.

Você pode manipular o evento keydown com o manipulador `onkeydown` de eventos.

O evento Keyup (onkeyup)

O evento keyup ocorre quando o usuário solta uma tecla no teclado.

Você pode manipular o evento keyup com o manipulador `onkeyup` de eventos.

O evento Keypress (onkeypress)

O evento de pressionamento de tecla ocorre quando um usuário pressiona uma tecla no teclado que possui um valor de caractere associado a ela. Por exemplo, teclas como Ctrl, Shift, Alt, Esc, teclas de seta, etc. não irão gerar um evento de pressionamento de tecla, mas irão gerar um evento de tecla e tecla.

Você pode manipular o evento de pressionamento de tecla com o manipulador `onkeypress` de eventos.

Eventos de formulário

Um evento de formulário é disparado quando um controle de formulário recebe ou perde o foco ou quando o usuário modifica um valor de controle de formulário, como digitando um texto em uma entrada de texto, selecionando qualquer opção em uma caixa de seleção, etc. Aqui estão alguns eventos de formulário mais importantes e seu manipulador de eventos.

O Evento Focus (onfocus)

O evento de foco ocorre quando o usuário dá foco a um elemento em uma página da web.

Você pode manipular o evento de foco com o manipulador `onfocus` de eventos.

O evento de desfoque (onblur)

O evento de desfoque ocorre quando o usuário tira o foco de um elemento de formulário ou janela.

Você pode manipular o evento blur com o manipulador `onblur` de eventos.

O Evento de Mudança (onchange)

O evento de mudança ocorre quando um usuário altera o valor de um elemento do formulário.

Você pode manipular o evento de mudança com o manipulador `onchange` de eventos.

O evento de envio (onsubmit)

O evento de envio ocorre apenas quando o usuário envia um formulário em uma página da web.

Você pode manipular o evento submit com o manipulador `onsubmit` de eventos.

Eventos de documento / janela

Os eventos também são acionados em situações quando a página é carregada ou quando o usuário redimensiona a janela do navegador, etc. Aqui estão alguns eventos de documentos / janelas mais importantes e seu manipulador de eventos.

O evento de carregamento (onload)

O evento de carregamento ocorre quando uma página da web termina de carregar no navegador da web.

Você pode manipular o evento de carregamento com o manipulador `onload` de eventos.

O evento de descarregamento (onunload)

O evento de descarregamento ocorre quando um usuário sai da página da web atual.

Você pode manipular o evento unload com o manipulador `onunload` de eventos.

O evento de redimensionamento (onresize)

O evento de redimensionamento ocorre quando um usuário redimensiona a janela do navegador. O evento de redimensionamento também ocorre em situações em que a janela do navegador é minimizada ou maximizada.

Você pode manipular o evento resize com o manipulador `onresize` de eventos.

JavaScript Strings

Uma string é uma sequência de letras, números, caracteres especiais e valores aritméticos ou combinação de todos. Strings podem ser criados colocando o literal de string (ou seja, caracteres de string) entre aspas simples (') ou aspas duplas (").

Sequências de escape de JavaScript

As sequências de escape também são úteis para situações em que você deseja usar caracteres que não podem ser digitados em um teclado. Aqui estão algumas outras sequências de escape mais comumente usadas.

- `\n` é substituído pelo caractere de nova linha
- `\t` é substituído pelo caractere de tabulação
- `\r` é substituído pelo caractere de retorno de carro
- `\b` é substituído pelo caractere backspace
- `\\`é substituído por uma única barra invertida (`\`)

Executando Operações em Strings

JavaScript fornece várias propriedades e métodos para realizar operações em valores de string. Tecnicamente, apenas objetos podem ter propriedades e métodos. Mas, em JavaScript, os tipos de dados primitivos podem agir como objetos quando você se refere a eles com a notação de acesso à propriedade (ou seja, notação de ponto).

JavaScript possibilitando a criação de um objeto wrapper temporário para tipos de dados primitivos. Esse processo é feito automaticamente pelo interpretador JavaScript em segundo plano.

Obtendo o comprimento de uma string

A propriedade length retorna o comprimento da string, que é o número de caracteres contidos na string. Isso inclui também o número de caracteres especiais, como `\t` ou `\n`.

Encontrando uma string dentro de outra string

Você pode usar o método `indexOf()`para localizar uma substring ou string dentro de outra string. Este método retorna o índice ou posição da primeira ocorrência de uma string especificada dentro de uma string.

Da mesma forma, você pode usar o método `lastIndexOf()`para obter o índice ou a posição da última ocorrência da string especificada dentro de uma string.

Os métodos `indexOf()`, e `lastIndexOf()` retornam `-1` se a substring não for encontrada. Ambos os métodos também aceitam um parâmetro opcional inteiro que especifica a posição dentro da string na qual iniciar a pesquisa.

Procurando um padrão dentro de uma string

Você pode usar o método `search()`para pesquisar uma parte específica do texto ou padrão dentro de uma string.

Como o método `indexOf()`, o método `search()` também retorna o índice da primeira correspondência e retorna `-1`se nenhuma correspondência for encontrada, mas, ao contrário do método `indexOf()`, esse método também pode ter uma *expressão regular* como seu argumento para fornecer recursos de pesquisa avançada.

Extraindo uma substring de uma string

Você pode usar o método `slice()`para extrair uma parte ou substring de uma string.

Este método leva 2 parâmetros: *índice inicial* (índice no qual iniciar a extração) e um *índice final* opcional (índice antes do qual encerrar a extração), como `str.slice(startIndex, endIndex)`.

Você também pode especificar valores negativos. O valor negativo é tratado como `strLength + startIndex`, onde `strLength` é o comprimento da cadeia (ou seja `str.length`), por exemplo, se `startIndex`é `-5` que é tratado como `strLength - 5`. Se `startIndex`for maior ou igual ao comprimento da string, o método `slice()`retorna uma string vazia. Além disso, se opcional `endIndex` não for especificado ou omitido, o método `slice()`será extraído até o final da string.

Você também pode usar o método `substring()`para extrair uma seção de uma determinada string com base nos índices inicial e final, como `str.substring(startIndex, endIndex)`. O método `substring()`é muito semelhante ao método `slice()`, exceto algumas diferenças:

- Se um dos argumentos for menor que `0` ou for `NaN`, ele será tratado como `0`.
- Se qualquer um dos argumentos for maior que `str.length`, ele será tratado como se fosse `str.length`.
- Se `startIndex` for maior que `endIndex`, então `substring()`irá trocar esses dois argumentos; por exemplo `str.substring(5, 0) == str.substring(0, 5)`

Extração de um número fixo de caracteres de uma string

JavaScript também fornece o método `substr()` que é semelhante a, `slice()` com uma diferença sutil, o segundo parâmetro especifica o número de caracteres a serem extraídos em vez do índice final, como `str.substr(startIndex, length)`. Se `length` for `0` ou um número negativo, uma string vazia será retornada.

Substituindo o conteúdo de uma string

Você pode usar o método `replace()`para substituir parte de uma string por outra string. Este método usa dois parâmetros: uma expressão regular para corresponder ou substring a ser substituída e uma string de substituição, como `str.replace(regexp|substr, newSubstr)`.

Este método `replace()` retorna uma nova string, não afeta a string original que permanecerá inalterada.

Por padrão, o método `replace()` substitui apenas a primeira correspondência e faz distinção entre maiúsculas e minúsculas. Para substituir a substring dentro de uma string sem fazer distinção entre maiúsculas e minúsculas, você pode usar uma *expressão regular (regexp)* com um modificador `i`, da mesma forma, para substituir todas as ocorrências de uma substring dentro de uma string sem fazer distinção entre maiúsculas e minúsculas, você pode usar o modificador `g` junto com o modificador `i`.

Converter uma string em maiúsculas ou minúsculas

Você pode usar o método `toUpperCase()` para converter uma string em **maiúsculas**. Da mesma forma, você pode usar o toLowerCase() para converter uma string em **minúsculas**.

Acessando caracteres individuais de uma string

Você pode usar o método `charAt()` para acessar caracteres individuais de uma string, como `str.charAt(index)`. O especificado `index` deve ser um número inteiro entre `0` e `str.length - 1`.
Se nenhum índice for fornecido, o primeiro caractere da string será retornado, pois o padrão é `0`.

Existe uma maneira ainda melhor de fazer isso. Desde ECMAScript 5, as strings podem ser tratadas como matrizes somente leitura, e você pode acessar caracteres individuais de uma string usando colchetes (`[]`) em vez do método `charAt()`.

Dividindo uma String em uma Matriz

O método `split()`pode ser usado para dividir uma string em uma matriz de strings, usando a sintaxe `str.split(separator, limit)`. O argumento `seperator` especifica a string em que cada divisão deve ocorrer, enquanto os argumentos `limit` especificam o comprimento máximo da matriz.

Se o argumento `separator` for omitido ou não for encontrado na string especificada, toda a string será atribuída ao primeiro elemento da matriz.

Números de JavaScript

Trabalhando com Números

JavaScript suporta números inteiros e de ponto flutuante que podem ser representados em notação decimal, hexadecimal ou octal. Ao contrário de outras linguagens, o JavaScript não trata números inteiros e números de ponto flutuante de maneira diferente. Todos os números em JavaScript são representados como números de ponto flutuante.

Os números também podem ser representados em notação hexadecimal (base 16). Os números hexadecimais são prefixados com `0x`. Eles são comumente usados para representar cores .

Evitando problemas de precisão

Às vezes, as operações em números de ponto flutuante produzem resultados inesperados. Essa diferença é chamada de ***erro de representação*** ou ***erro de arredondamento***. Isso ocorre porque o JavaScript e muitas outras linguagens usam a forma binária (base 2) para representar os números decimais (base 10) internamente. Infelizmente, a maioria das frações decimais não pode ser representada exatamente na forma binária, então pequenas diferenças ocorrem.

Execução de operações em números

JavaScript fornece várias propriedades e métodos para realizar operações em valores numéricos. Como você já sabe dos capítulos anteriores, em JavaScript, os tipos de dados primitivos podem agir como objetos quando você se refere a eles com a notação de acesso à propriedade (ou seja, notação de ponto).

Nas seções a seguir, veremos os métodos de número mais comumente usados.

Analisando inteiros de strings

O método `parseInt()` pode ser usado para analisar um inteiro de uma string. Este método é particularmente útil em situações em que você está lidando com os valores como unidades CSS por exemplo `50px`, `12pt`, etc, e você gostaria de extrair o valor numérico fora dele.

Se o método `parseInt()` encontrar um caractere que não seja numérico na base especificada, ele interromperá a análise e retornará o valor inteiro analisado até aquele ponto. Se o primeiro caractere não puder ser convertido em um número, o método retornará `NaN`(não um número).

Espaços à esquerda e à direita são permitidos.

Da mesma forma, você pode usar o método `parseFloat()` para analisar o número de ponto flutuante de uma string. O método `parseFloat()` funciona da mesma maneira que o método `parseInt()`, exceto que ele recupera inteiros e números com decimais.

Convertendo Números em Strings

O método `toString()` pode ser usado para converter um número em seu equivalente de string. Este método aceita opcionalmente um parâmetro inteiro no intervalo de 2 a 36 especificando a base a ser usada para representar valores numéricos.

Formatando Números para Decimais Fixos

Você pode usar o método `toFixed()`quando quiser formatar um número com um número fixo de dígitos à direita da vírgula decimal. O valor retornado por este método é uma string e tem um número especificado exatamente `digits` após a vírgula decimal. Se o parâmetro `digits` não for especificado ou omitido, ele será tratado como 0

Formatando Números com Precisão

Se quiser a forma mais apropriada de um número, você pode usar o método `toPrecision()`. Este método retorna uma string que representa o número com a precisão especificada.

Se a precisão for grande o suficiente para incluir todos os dígitos da parte inteira do número, o número será formatado usando a notação de ponto fixo. Caso contrário, o número é formatado em notação exponencial. O parâmetro de precisão é opcional

Encontrando os maiores e menores números possíveis

O objeto Number também possui várias propriedades associadas a ele. As propriedades `Number.MAX_VALUE` e `Number.MIN_VALUE`do objeto Number representam os maiores e menores (mais próximos de zero, não os mais negativos) números positivos possíveis que o JavaScript pode manipular. Eles são constantes e seus valores reais são `1.7976931348623157e+308`, e `5e-324`, respectivamente.

Um número que está fora do intervalo de números possíveis é representado por uma constante `Number.POSITIVE_INFINITY` ou `Number.NEGATIVE_INFINITY`.

Declarações JavaScript If ... Else

Como muitas outras linguagens de programação, JavaScript também permite escrever código que executa ações diferentes com base nos resultados de condições de teste lógico ou comparativo em tempo de execução. Isso significa que você pode criar condições de teste na forma de expressões que avaliam `true` ou `false` e, com base nesses resultados, você pode executar determinadas ações.

Existem várias instruções condicionais em JavaScript que você pode usar para tomar decisões:

- A declaração **if**
- A instrução **if ... else**
- A instrução **if ... else if .... else**
- A instrução **switch ... case**

A declaração “if”

A instrução *if* é usada para executar um bloco de código apenas se a condição especificada for avaliada como verdadeira.

A declaração – if / else

Você pode aprimorar os recursos de tomada de decisão de seu programa JavaScript, fornecendo uma escolha alternativa por meio da adição de uma instrução *else* à instrução *if* .

A instrução *if … else* permite que você execute um bloco de código se a condição especificada for avaliada como verdadeira e outro bloco de código se for avaliada como falsa.

A declaração if / elseif / else

O *if … else if … else* uma instrução especial que é usada para combinar várias instruções *if … else*.

O operador ternário

O operador ternário fornece uma maneira abreviada de escrever as instruções *if … else*. O operador ternário é representado pelo `?`símbolo de interrogação ( ) e leva três operandos: uma condição para verificar, um resultado para `true` eum resultado para `false`.

Switch...Case

Usando o Switch /Case Statement

A instrução *switch/case* é uma alternativa à instrução *if … else if … else* , que faz quase a mesma coisa. A instrução *switch … case* testa uma variável ou expressão em relação a uma série de valores até encontrar uma correspondência e, em seguida, executa o bloco de código correspondente a essa correspondência.

A instrução *switch ... case* difere da instrução *if ... else de* uma maneira importante. A instrução switch é executada linha por linha (ou seja, instrução por instrução) e uma vez que o JavaScript encontra uma cláusula `case` avaliada como verdadeira, ele não apenas executa o código correspondente a essa cláusula de caso, mas também executa todas as cláusulas `case` subsequentes até o final do bloco `switch` automaticamente .

Para evitar isso, você deve incluir uma declaração `break` após cada `case`(como você pode ver no exemplo acima). A instrução `break` diz ao interpretador JavaScript para sair do bloco de instrução *switch ... case*, uma vez que executa o código associado ao primeiro caso verdadeiro.

A instrução `break`, entretanto, não é necessária para a cláusula `case` ou `default`, quando ela finalmente aparece em uma instrução switch. Embora seja uma boa prática de programação encerrar a última cláusula `case` ou `defaul t`em uma instrução switch com um `break`. Isso evita um possível erro de programação posteriormente, se outra instrução case for adicionada à instrução switch.

A cláusula `default` é opcional, que especifica as ações a serem realizadas se nenhuma corresponder `case` à expressão switch. A cláusula `default` não precisa ser a última cláusula a aparecer em uma instrução switch. Aqui está um exemplo, onde `default` não é a última cláusula.

Vários casos compartilhando a mesma ação

Cada valor de caso deve ser exclusivo em uma instrução switch. No entanto, casos diferentes não precisam ter uma ação única.

O que é o Angular?

Aqui quero te explicar de forma bem sucinta a entender o que é o Angular, quais as vantagens que ele oferece e o que você pode esperar ao começar a construir seus aplicativos.

Angular é uma plataforma de desenvolvimento, construída em TypeScript. Como plataforma, o Angular inclui:

- Uma estrutura baseada em componentes para a construção de aplicativos da web escalonáveis
- Uma coleção de bibliotecas bem integradas que cobrem uma ampla variedade de recursos, incluindo roteamento, gerenciamento de formulários, comunicação cliente-servidor e muito mais
- Um conjunto de ferramentas de desenvolvedor para ajudá-lo a desenvolver, construir, testar e atualizar seu código

Com o Angular, você aproveitará a vantagem de uma plataforma que pode ser dimensionada de projetos de um único desenvolvedor a aplicativos de nível empresarial. O Angular foi projetado para tornar a atualização o mais fácil possível, para que você possa aproveitar as vantagens dos desenvolvimentos mais recentes com um mínimo de esforço. O melhor de tudo é que o ecossistema Angular consiste em um grupo diversificado de mais de 1,7 milhões de desenvolvedores, autores de bibliotecas e criadores de conteúdo.

Aplicações angulares: o essencial

Agora vamos explicar as ideias principais por trás do Angular. Compreender essas ideias pode ajudá-lo a projetar e construir seus aplicativos de forma mais eficaz.

Componentes

Os componentes são os blocos de construção que compõem um aplicativo. Um componente inclui uma classe TypeScript com um decorator (@Component), um modelo HTML e estilos. O @Component especifica as seguintes informações específicas do Angular

- Um seletor CSS que define como o componente é usado em um modelo. Os elementos HTML em seu modelo que correspondem a este seletor tornam-se instâncias do componente.
- Um modelo HTML que instrui o Angular como renderizar o componente.

Um conjunto opcional de estilos CSS que definem a aparência dos elementos HTML do modelo.

O exemplo abaixo é um modelo básico de um @Component.

```
import { Component } from '@angular/core';

@Component({
  selector: 'hello-world',
  template: `
    <h2>Hello World</h2>
    <p>This is my first component!</p>
    `,
})
export class HelloWorldComponent {
  // The code in this class drives the component's behavior.
}
```

Figura 138 – Angular: Componente básico

Para utilizar este componente, basta chama-lo da seguinte forma num template HTML:

```
<hello-world></hello-world>
```

Figura 139 - Angular: Chamando um componente num template HTML

Quando o Angular renderiza este componente, o resultado no navegador é algo parecido a isso:

```
<hello-world>
    <h2>Hello World</h2>
    <p>This is my first component!</p>
</hello-world>
```

Figura 140 - Angular: Resultado no navegador de um componente renderizado

O modelo de componente do Angular oferece forte encapsulamento e uma estrutura de aplicativo intuitiva. Os componentes também tornam seu aplicativo mais fácil de testar a unidade e podem melhorar a legibilidade geral de seu código.

Modelos

Cada componente possui um template HTML que declara como aquele componente é renderizado. Você define este modelo embutido ou por caminho de arquivo.

O Angular estende HTML com sintaxe adicional que permite inserir valores dinâmicos de seu componente. O Angular atualiza automaticamente o DOM renderizado quando o estado do seu componente muda. Uma aplicação desse recurso é a inserção de texto dinâmico, conforme mostrado no exemplo a seguir.

```
<p>{{ message }}</p>
```

Figura 141 - Angular: Exibindo o valor de uma variável definida no modelo de negócios

O valor da mensagem vem da classe do componente:

```
import { Component } from '@angular/core';

@Component ({
  selector: 'hello-world-interpolation',
  templateUrl: './hello-world-interpolation.component.html'
})
export class HelloWorldInterpolationComponent {
    message = 'Hello, World!';
}
```

Figura 142 - Angular: Declarando variável para exibir no template HTML

Quando o aplicativo carrega o componente e seu modelo, o usuário vê o seguinte:

```
<p>Hello, World!</p>
```

Figura 143 - Angular: Resultado da variável para o usuário

Observe o uso de chaves duplas - elas instruem o Angular a interpolar o conteúdo dentro delas.

Property Bindings

O Angular também oferece suporte a associações de propriedade (*property binding*), para ajudá-lo a definir valores para propriedades e atributos de elementos HTML e passar valores para a lógica de apresentação de seu aplicativo.

```
<p [id]="sayHelloId" [style.color]="fontColor">You can set my color in the component!
</p>
```

Figura 144 - Angular: Aplicando property bindings

Observe o uso de colchetes - essa sintaxe indica que você está associando a propriedade ou atributo a um valor na classe do componente.

Event Listeners

Você também pode declarar "ouvintes de eventos" (*event listener*) para ouvir e responder às ações do usuário, como pressionamentos de tecla, movimentos do mouse, cliques e toques. Você declara um ouvinte de evento especificando o nome do evento entre parênteses:

```
<button (click)="sayMessage()" [disabled]="canClick">Trigger alert message</button>
```

Figura 145 - Angular: Ouvindo eventos que são acionados pelo usuário

O exemplo acima chama um método que é definido na classe do Componente.

```
sayMessage() {
    alert(this.message);
}
```

Figura 146 - Angular: Método declarado no Component que será acionado pelo Event Listener

A seguir está um exemplo de interpolação e ligações dentro de um modelo Angular:

```
1. import { Component } from '@angular/core';
2.
3. @Component ({
4.   selector: 'hello-world-bindings',
5.   templateUrl: './hello-world-bindings.component.html'
6. })
7. export class HelloWorldBindingsComponent {
8.     fontColor = 'blue';
9.     sayHelloId = 1;
10.    canClick = false;
11.    message = 'Hello, World';
12.    sayMessage() {
13.        alert(this.message);
14.    }
15. }
```

Figura 147 - Angular: Exemplo de componente com método declarado

Enquanto isto, no template HTML, toda a página se resumiria a algo como:

```
<button (click)="sayMessage()" [disabled]="canClick">Trigger alert message</button>
<p [id]="sayHelloId" [style.color]="fontColor">You can set my color in the component!
</p>
```

Figura 148 - Angular: Exemplo de template HTML com event listener que será chamado pelo Component

Diretivas

Você pode adicionar funcionalidades adicionais aos seus modelos por meio do uso de diretivas. As diretivas mais populares em Angular são ***ngIf** e ***ngFor**.

Você pode usar diretivas para realizar uma variedade de tarefas, como modificar dinamicamente a estrutura do DOM. Você também pode criar suas próprias diretivas personalizadas para criar excelentes experiências de usuário.

O código a seguir é um exemplo da diretiva.

```
1. import { Component } from '@angular/core';
2.
3. @Component({
4.     selector: 'hello-world-ngif',
5.     templateUrl: './hello-world-ngif.component.html'
6.   })
7. export class HelloWorldNgIfComponent {
8.   message = 'I\'m read only!';
9.   canEdit = false;
10.
11.   onEditClick() {
12.     this.canEdit = !this.canEdit;
13.     if (this.canEdit) {
14.       this.message = 'You can edit me!';
15.     } else {
16.       this.message = 'I\'m read only!';
17.     }
18.   }
19. }
```

Figura 149 - Angular: Exemplo de diretiva

Os modelos declarativos do Angular permitem separar claramente a lógica do seu aplicativo de sua apresentação. Os modelos são baseados em HTML padrão, portanto, são fáceis de construir, manter e atualizar.

Injeção de dependência

A injeção de dependência permite declarar as dependências de suas classes TypeScript sem cuidar de sua instanciação. O Angular trata da instanciação para você. Este padrão de design permite que você escreva um código mais testável e flexível. Embora entender a injeção de dependência não seja crítica para começar a usar o Angular, é altamente recomendável que

você tenha em mente a importância de compreender a aplicação disto, pois ela é uma prática recomendada em muitos aspectos do Angular e o framework tira proveito deste recurso em vários pontos.

Para ilustrar como a injeção de dependência funciona, considere o exemplo a seguir. O primeiro arquivo **logger.service.ts**, define uma classe **Logger**. Esta classe contém uma função **writeCount** que exibe um número no console.

```
import { Injectable } from '@angular/core';

@Injectable({providedIn: 'root'})
export class Logger {
  writeCount(count: number) {
    console.warn(count);
  }
}
```

Figura 150 - Angular: Exemplo da logger.service.ts

Em seguida, o arquivo **hello-world-di.component.ts** define um componente angular. Este componente contém um botão que usa a função **writeCount** da classe **Logger**. Para acessar essa função, o **LoggerService** é injetado na classe **HelloWorldDI** adicionando: ***"private logger: Logger"*** ao construtor.

```
import { Component } from '@angular/core';
import { Logger } from '../logger.service';

@Component({
  selector: 'hello-world-di',
  templateUrl: './hello-world-di.component.html'
})
export class HelloWorldDependencyInjectionComponent  {
  count = 0;

  constructor(private logger: Logger) {
  }

  onLogMe() {
    this.logger.writeCount(this.count);
    this.count++;
  }
}
```

Figura 151 - Angular: Exemplo de Injeção de Dependência

Angular CLI

O Angular CLI é a maneira mais rápida, fácil e recomendada de desenvolver aplicativos Angular. O Angular CLI facilita várias tarefas. aqui estão alguns exemplos:

Tabela 3 - Angular: Comandos do Angular CLI

ng build	Compila um aplicativo Angular em um diretório de saída.
ng serve	Constrói e atende seu aplicativo, reconstruindo com base nas alterações do arquivo.
ng generate	Gera ou modifica arquivos com base em um esquema.
ng test	Executa testes de unidade em um determinado projeto.
ng e2e	Constrói e fornece um aplicativo Angular e, em seguida, executa testes de ponta a ponta.

Você descobrirá que o Angular CLI é uma ferramenta valiosa para desenvolver seus aplicativos.

Bibliotecas primárias

Os benefícios do Angular realmente se tornam aparentes quando seu aplicativo começa a crescer e você deseja adicionar funções adicionais, como navegação no site ou login do usuário. É aí que você pode aproveitar a plataforma Angular para incorporar uma das muitas bibliotecas primárias que o Angular oferece.

Algumas das bibliotecas disponíveis para você incluem:

Angular Router	Navegação avançada do lado do cliente e roteamento com base em componentes angulares. Suporta carregamento lento, rotas aninhadas, correspondência de caminho personalizado e muito mais.
Angular Forms	Sistema uniforme de participação e validação de formulários.
Angular HttpClient	Cliente HTTP robusto que pode possibilitar uma comunicação cliente-servidor mais avançada.
Angular Animations	Sistema rico para conduzir animações com base no estado do aplicativo.
Angular PWA	Ferramentas para a construção de Progressive Web Applications (PWAs), incluindo um service worker e manifesto do Web app.
Angular Schematics	Ferramentas automatizadas de scaffolding, refatoração e atualização que simplificam o desenvolvimento em grande escala.

Essas bibliotecas expandem a funcionalidade de seu aplicativo e, ao mesmo tempo, permitem que você se concentre mais nos recursos que tornam seu aplicativo único. E você pode adicionar essas bibliotecas sabendo que elas foram projetadas para se integrar perfeitamente e atualizar simultaneamente com a estrutura Angular.

Essas bibliotecas são necessárias apenas se, e quando puderem ajudá-lo a adicionar funcionalidade aos seus aplicativos ou resolver um problema específico.