



Co-funded by the
Erasmus+ Programme
of the European Union



3.1.:

Pacote de Material de Formação Code4SP

WP3:

Materiais de Formação
Code4SP

Preparado por:



Informação do Projeto

Acrónimo do projeto: Code4SP

Título do projeto: Coding for Social Promotion

Referência do projeto: 621417-EPP-1-2020-1-PT-EPPKA3-IPI-SOC-IN

Website do projeto: www.code4sp.eu

Parceiro autor: CEPROF

Versão do documento: 1

Data de preparação: 11/02/2022

| História do Documento | | | |
|-----------------------|--------|--------|-----------|
| Data | Versão | Autor | Descrição |
| 11/02/2022 | 1 | CEPROF | Esboço |
| | | | |
| | | | |
| | | | |

Tabela de Conteúdos

| | |
|-----------------------------------|-------------------------------------|
| Informação do Projeto..... | Error! Bookmark not defined. |
| Tópico:..... | 5 |
| 2. HTML..... | 5 |
| Pré-requisitos:..... | 5 |
| Carga de trabalho:..... | 5 |
| Descrição:..... | 5 |
| Resultados da aprendizagem:..... | 5 |
| Material necessário:..... | 5 |
| Cenário de aula:..... | 6 |
| Subtópicos:..... | 6 |
| Recursos adicionais:..... | 6 |
| 2.1. As bases de HTML..... | 7 |
| Anatomia de um elemento HTML..... | 7 |
| Navegadores de Web..... | Error! Bookmark not defined. |
| Estrutura de página HTML..... | 9 |
| História HTML..... | 10 |
| Editores HTML..... | 10 |
| Exemplos básicos HTML..... | 12 |
| Atributos HTML..... | 16 |
| Títulos HTML..... | 19 |
| Parágrafos HTML..... | 20 |
| Estilos HTML..... | 23 |
| Formatação de texto HTML..... | 25 |
| Formatação de texto HTML..... | 27 |
| Comentários HTML..... | Error! Bookmark not defined. |
| Cores HTML..... | Error! Bookmark not defined. |
| Ligações HTML..... | Error! Bookmark not defined. |
| Imagens HTML..... | 38 |
| Ícones HTML..... | 40 |
| Tabelas HTML..... | Error! Bookmark not defined. |

| | |
|---|-------------------------------------|
| Listas HTML..... | 46 |
| Formulários HTML | Error! Bookmark not defined. |
| iFrame HTML..... | 54 |
| 2.2. Conceitos avançados HTML..... | 56 |
| Doctypes HTML | Error! Bookmark not defined. |
| Esquemas HTML | Error! Bookmark not defined. |
| Cabeçalho HTML..... | Error! Bookmark not defined. |
| O elemento base HTML..... | 59 |
| O elemento de ligação HTML..... | Error! Bookmark not defined. |
| O elemento de estilo HTML | Error! Bookmark not defined. |
| O elemento meta HTML..... | 61 |
| O elemento roteiro HTML..... | Error! Bookmark not defined. |
| Entidades HTML | Error! Bookmark not defined. |
| URL HTML..... | 66 |
| Validação HTML | 69 |
| Características HTML5 | 71 |
| Novos tipos de entrada HTML5..... | Error! Bookmark not defined. |
| Canvas HTML5 | 79 |
| SVG HTML5 | 86 |
| Áudio HTML 5..... | 91 |
| Vídeo HTML5..... | 92 |
| Armazenamento Web HTML5..... | 94 |
| Cache de aplicação HTML5..... | Error! Bookmark not defined. |
| Trabalhadores Web HTML5..... | Error! Bookmark not defined. |
| Eventos enviados pelo servidor HTML5..... | Error! Bookmark not defined. |
| Geolocalização HTML 5..... | 105 |
| Arrastar e soltar HTML5..... | Error! Bookmark not defined. |
| Referências HTML5 | 110 |

Tópico:

2. HTML

Pré-requisitos:

Literacia básica de computadores, software básico instalado e conhecimento básico de trabalhar com ficheiros.

Carga de trabalho:

10 horas.

Descrição:

Neste tópico, vamos cobrir os básicos de HTML, para atualizar os alunos no mundo da programação, depois de adquirir alguns conceitos básicos. Definimos os elementos, atributos e todos os outros termos importantes que eles podem ter ouvido e onde esses termos se encaixam na linguagem. Também mostramos como um elemento HTML é estruturado, como uma página HTML típica é estruturada e explicamos outros recursos básicos importantes da linguagem.

Resultados da aprendizagem:

- Reconhecer o conceito de HyperText Markup Language (HTML) na família de linguagens de descrição de documentos.
- Distinguir entre estrutura, conteúdo e estilos de página.
- Usar HTML na construção de páginas para a Web.

Material necessário:

- Computador ou portátil
- Conexão à Internet
- Construtor online de Websites (<https://sites.google.com/new>)
- Editor online de texto (<https://www.w3schools.com/html/default.asp>)

Cenário de Aula:

O tempo total para este tópico é de 10 horas e caberá ao formador/coach decidir quanto tempo dedicar ao ensino de cada subtópico. Para aproveitar ao máximo todo o tempo disponível, propomos a utilização dos materiais de formação produzidos pelo projeto (apresentações PPT), que foram concebidos a pensar numa utilização eficaz do tempo. Estas apresentações são compostas pelos seguintes elementos:

- Desenvolvimento do subtópico e principais ideias a reter;
- Atividades/Exercícios propostos.

Dito isto, se o formador/coach seguir a sequência lógica dos PPTs, certamente conseguirá completar a sessão dentro do prazo estipulado. Essas apresentações também podem ser disponibilizadas aos alunos para estudo individual.

Subtópicos:

- 2.1. As bases de HTML
- 2.2. Conceitos avançados de HTML
- 2.3. Características de HTML5
- 2.4. Referências de HTML5

Recursos adicionais:

- [HTML reference guide](#)
- [W3Schools](#) - guia para cada elemento HTML e regra CSS, e exemplos para cada um deles
- [Khan Academy](#): recursos e vídeos úteis sobre codificação HTML e CSS, em vários idiomas

2.1. As bases de HTML

HTML (Hypertext Markup Language) **não é** uma linguagem de programação. É uma linguagem de marcação que comunica aos navegadores da web como estruturar as páginas da web visitadas. Pode ser tão complexo ou tão simples quanto o desenvolvedor web deseje que seja. HTML compreende uma série de elementos, usados para incluir, envolver ou elevar diferentes partes do conteúdo, para o fazer aparecer ou agir de uma determinada maneira. As tags anexas podem transformar o conteúdo num hyperlink para conectar a outra página, colocar palavras em itálico e assim por diante. Por exemplo, considerando a seguinte linha de texto:

```
HTML is cool.
```

Figura 1 – Linha de texto "HTML is cool" (Fonte: Autor)

Se alguém quisesse que o texto ficasse sozinho, poderia especificar que é um parágrafo, colocando-o num elemento de parágrafo (<p>):

```
<p> HTML is cool.</p>
```

Figura 2 – Codificação para o parágrafo "HTML is cool" (Fonte: Autor)

Nota: Tags em HTML não diferenciam maiúsculas de minúsculas. Isso significa que eles podem ser escritos em maiúsculas ou minúsculas. Por exemplo, uma tag <title> pode ser escrita como <title>, <TITLE>, <Title>, <TiTIE>, etc., e funcionará. No entanto, é uma prática recomendada escrever todas as tags em letras minúsculas para efeitos de consistência e legibilidade.

Anatomia do elemento HTML

Vamos explorar mais o elemento de parágrafo da secção anterior:

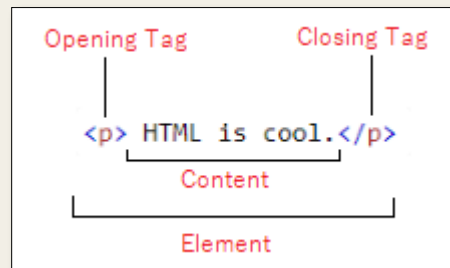


Figura 3 – Anatomia de um elemento HTML (Fonte: Autor)

Portanto, a anatomia do elemento é composta por:

A tag de abertura: o nome do elemento (neste exemplo, p para parágrafo), entre sinais angulares de abertura e fecho. Essa tag de abertura marca onde o elemento começa ou começa a ter efeito. Neste exemplo, ele vem primeiro, no início do texto do parágrafo.

O conteúdo: Isto é o conteúdo do elemento. Neste exemplo, é o texto do parágrafo.

A tag de fecho: É o mesmo que a tag de abertura, exceto que inclui uma barra antes do nome do elemento. Isso marca onde o elemento termina. Não incluir uma tag de fecho é um erro comum para iniciantes que pode produzir resultados peculiares.

O **elemento** é a tag de abertura, seguido pelo conteúdo, seguido pela tag de fecho.

Nota: Alguns elementos HTML não possuem conteúdo (como o elemento
). Esses elementos são chamados de “elementos vazios”. Eles não têm uma tag final!

Navegadores Web

O objetivo de um navegador da Web (Chrome, Edge, Firefox, Safari) é ler documentos HTML e exibi-los corretamente.

O navegador não mostra as tags HTML. Usa-as para determinar como exibir o documento:



Figura 4 – Um documento HTML determinado num navegador web (Fonte: https://www.w3schools.com/html/html_intro.asp)

Estrutura da página HTML

Uma página HTML deve ser estruturada da seguinte forma:

```

<html>
  <head>
    <title>Page title</title>
  </head>
  <body>
    <h1>This is a heading</h1>
    <p>This is a paragraph.</p>
    <p>This is another paragraph.</p>
  </body>
</html>

```

Figura 5 – Estrutura de uma página HTML (Fonte: https://www.w3schools.com/html/html_intro.asp)

O conteúdo dentro da secção <body> (a área branca acima) será exibido num navegador. O conteúdo dentro do elemento <title> será mostrado na barra de título do navegador ou na aba da página.

História do HTML

Desde os primeiros dias da World Wide Web, já existiram várias versões de HTML:

| Ano | Versão |
|--------|--|
| 1989 | Tim Berners-Lee inventou www |
| 1991 | Tim Berners-Lee inventou HTML |
| 1993 | Dave Raggett esboçou HTML+ |
| 1995 | HTML Working Group definiu HTML 2.0 |
| 1997 | W3C Recomendação: HTML 3.2 |
| 1999 | W3C Recomendação: HTML 4.01 |
| 2000 | W3C Recomendação: XHTML 1.0 |
| 2008 | WHATWG HTML5 Primeiro esboço público |
| 2012 | WHATWG HTML5 Padrão de vida |
| 2014 | W3C Recomendação: HTML5 |
| 2016 | W3C Recomendação de candidatos: HTML 5.1 |
| 2017 | W3C Recomendação: HTML5.1 2nd Edition |
| 2017 - | W3C Recomendação: HTML5.2 |

Tabela 1 – História HTML (Fonte: https://www.w3schools.com/html/html_intro.asp)

Este manual segue o mais recente padrão HTML5.

Editores HTML

Um simples editor de texto é tudo o que é preciso para aprender HTML.

Aprender HTML Usando Notepad ou TextEdit

Páginas Web pode ser criadas e modificadas usando editores profissionais de HTML.

Contudo, para aprender HTML, um simples editor de texto com o Notepad (PC) ou o TextEdit (Mac) é recomendado. Usar um simples editor de texto pode ser uma boa forma de aprender HTML.

Os passos abaixo devem ser seguidos para criar a primeira página Web do aprendiz, usando o Notepad ou o TextEdit.

Passo 1: Abrir o Notepad (PC)

(Windows 8 ou mais recente: Abrir Iniciar (o símbolo de janela no canto inferior esquerdo do ecrã). Escrever Notepad.

Windows 7 ou mais antigo: Abrir Iniciar > Programas > Acessórios > Notepad)

- Abrir TextEdit (Mac)
- Abrir Procurar > Aplicações > TextEdit
- Fazer com que a aplicação grave os ficheiros da forma correta. Em Preferências > Formato > escolher "Plain Text"
- Depois, por baixo de "Open and Save", carregar na caixa que diz "Display HTML files as HTML code instead of formatted text".
- Depois, **abrir novo documento** para colocar o código.

Passo 2: Escrever HTML

- Escreva ou copie o seguinte Código de HTML para o NotePad:

```
<!DOCTYPE html>
```

```
<html>
<body>
<h1> My First Heading</h1>
<p>My first paragraph.</p>
</body>
</html>
```

Passo 3: Guardar a página HTML

- Guarde o ficheiro no seu computador.
- Selecione File > Save as no menu do NotePad.
- Identifique o ficheiro como "index.htm" e defina a codificação para UTF-8 (a codificação preferencial para ficheiros HTML).

Passo 4: Visualize a página HTML no seu navegador

- Abra o ficheiro HTML guardado no seu navegador preferido (duplo click no ficheiro ou click direito e escolher "Open with").

O resultado será semelhante à seguinte imagem:



Figura 6 – Um documento HTML determinado num navegador web (Fonte: https://www.w3schools.com/html/html_intro.asp)

Exemplo básicos de HTML

Nesta secção, alguns exemplos básicos de HTML serão partilhados.

Documentos HTML

Todos os documentos HTML devem começar com uma declaração tipo documento: `<!DOCTYPE html>`.

A declaração `<!DOCTYPE>` representa o tipo de documento e ajuda os navegadores a exibir páginas web de forma correta. Só deve aparecer uma vez, no topo da página (antes de quaisquer tags HTML). Não é sensível ao caso.

A declaração `<!DOCTYPE>` para HTML5 é: `<!DOCTYPE html>`

Cabeçalhos HTML

Cabeçalhos HTML são definidos com as tags `<h1>` até `<h6>`.

`<h1>` define o cabeçalho mais importante. `<h6>` define o cabeçalho menos importante:

```
<h1>This is heading 1</h1>
<h2>This is heading 2</h2>
<h3>This is heading 3</h3>
```

Figura 7 – Cabeçalhos HTML (Fonte: <https://www.w3schools.com/html>)

Parágrafos HTML

Parágrafos HTML são definidos com a tag `<p>`:

```
<p>This is a paragraph.</p>
<p>This is another paragraph.</p>
```

Figura 8 – Parágrafos HTML (Fonte: <https://www.w3schools.com/html>)

Ligações HTML

Ligações HTML são definidas com a tag `<a>`:

```
<a href="https://www.w3schools.com">This is a link</a>
```

Figura 9 – Ligação HTML (Fonte: <https://www.w3schools.com/html>)

O destino da ligação é específico no atributo `href`.

Atributos são usados para providenciar informação adicional acerca de elementos HTML.

Mais acerca dos atributos será ensinado mais à frente.

Imagens HTML

Imagens HTML são definidas com a tag ``.

O ficheiro fonte (`src`), texto alternativo (`alt`), largura e altura são dados como atributos:

```

```

Figura 10 – Imagens HTML (Fonte: <https://www.w3schools.com/html>)

Como ver uma fonte HTML?

Ver Código de fonte HTML:

- Click direito numa página HTML e selecione "View Page Source" (no Chrome), "View Source" (no Edge) ou semelhante noutros navegadores. Isto irá abrir uma janela que contém o código fonte da página HTML.

Inspecionar um elemento HTML:

- Clique com o botão direito do rato num elemento (ou numa área em branco) e escolha "Inspecionar" ou "Inspecionar Elemento" para ver de que elementos são

compostos (irá ver o HTML e o CSS). Pode também editar o HTML ou CSS dinamicamente no painel Elementos ou Estilos, que é aberto.

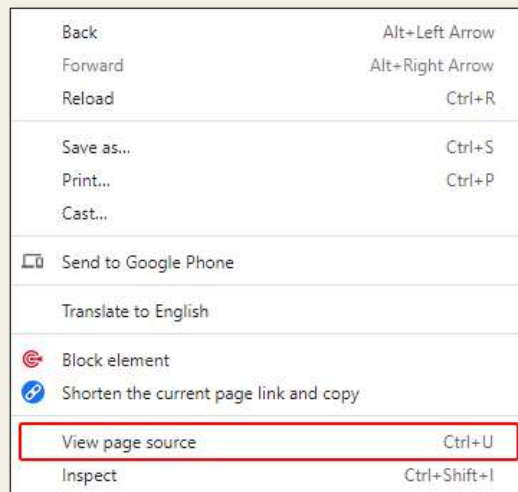


Figura 11 – Como ver a página fonte (Fonte:Author)

A estrutura de um documento HTML

O documento HTML começa com `<html>` e acaba com `</html>`. A parte visível do documento HTML está entre `<body>` e `</body>`.

```
<!DOCTYPE html>
<html>
<body>

<h1>My First Heading</h1>
<p>My first paragraph.</p>

</body>
</html>
```

Figura 12 – Declaração tipo documento, HTML e partes visíveis do documento HTML (Fonte: <https://www.w3schools.com/html>)

Atributos HTML

Os atributos HTML fornecem informações adicionais sobre elementos HTML e todos os elementos HTML podem tê-los. Os atributos fornecem informações adicionais sobre os elementos, sendo sempre especificados na tag inicial. Eles geralmente vêm em pares nome/valor como: `name="value"`.

Lista de atributos comuns:

- **href** – a tag `<a>` define um hyperlink. O atributo href especifica o URL da página para qual a ligação vai, da seguinte forma:

```
<a href="https://www.w3schools.com">Visit W3Schools</a>
```

- **src** – a tag `` é usada como uma imagem embutida numa página HTML. O atributo `<src>` especifica o caminho para a página ser exibida, como visto abaixo:

```

```

Existem duas formas de especificar o URL no atributo `<src>`:

1. **URL absoluto** – Liga a uma imagem externa que está situada num outro website. E.g.: `src="https://www.w3schools.com/images/img_girl.jpg"`.

Notas: Imagens externas podem estar sob **direitos de autor**. Se alguém não tiver permissão para usar, poderá estar a violar as leis dos direitos de autor. Além disso, as imagens externas não podem ser controladas; elas podem ser removidas ou alteradas abruptamente.

2. **URL relativo** - Liga a uma imagem situada no site. Aqui, o URL não inclui o nome do domínio. Se o URL começar sem uma barra, será relativo à página atual. E.g.: `src="img_girl.jpg"`. Se o URL começa com uma barra, será relativo ao domínio. E.g.: `src="/images/img_girl.jpg"`.

Dica: É quase sempre melhor usar URLs relativos. Eles não vão partir se o domínio mudar.

- **Atributos de largura e altura** – a tag `` deve também compreender os atributos de largura e altura, que estipula a largura e altura da imagem (em pixels):

```

```

- **alt** – o atributo alt necessário para a tag `` especifica um texto alternativo para uma imagem, se a imagem não puder ser exibida, por alguma razão. Isto pode ser devido a uma conexão lenta, um erro no atributo src, ou se o usuário usa um leitor de ecrã.

```

```

Se tentarmos exibir uma imagem que não existe, o valor do atributo alt será exibido, como segue:

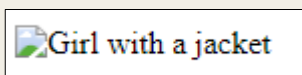


Figura 13 – Valor Alt se a imagem não existe (Fonte: <https://www.w3schools.com/html>)

- **Estilo** – o atributo estilo é usado para adicionar estilos a um elemento, como cor, fonte, tamanho e mais.

```
<p style="color:red;">This is a red paragraph.</p>
```

Resultado:

| | |
|---|---|
| <pre><!DOCTYPE html> <html> <body> <h2>The style Attribute</h2> <p>The style attribute is used to add styles to an element, such as color: </p> <p style="color:red;">This is a red paragraph.</p> </body> </html></pre> | <h2>The style Attribute</h2> <p>The style attribute is used to add styles to an element, such as color:</p> <p>This is a red paragraph.</p> |
|---|---|

Figura 14 – Código para colorir um parágrafo (Fonte: <https://www.w3schools.com/html>)

- **lang** – o atributo lang deve sempre estar incluído dentro da tag `<html>`, para declarar a linguagem da página Web. Isto destina-se a suportar motores de busca e navegadores. O exemplo abaixo estipula o inglês como o idioma em uso:

```
<!DOCTYPE html>
<html lang="en">
<body>
...
</body>
</html>
```

Os códigos de país também podem ser adicionados ao código de idioma no atributo **lang**. Assim, os dois primeiros caracteres expressam o **idioma** da página HTML e os dois últimos caracteres descrevem o **país**.

O exemplo a seguir especifica português como idioma e Portugal como país:

```
<!DOCTYPE html>
<html lang="pt-PT">
<body>
...
</body>
</html>
```

[HTML Language Code Reference](#) incluiu os códigos de todas as linguagens.

- **Título** – este atributo descreve algumas informações adicionais sobre um elemento. O seu valor será exibido como uma dica de ferramenta quando o ponteiro do rato passar sobre o elemento, como segue:

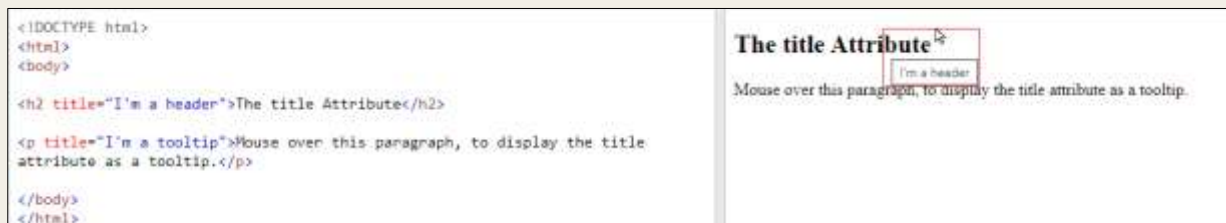


Figure 15 – Código para exibir uma dica de ferramenta 'título' (Fonte: Autor)

Recomendação: É altamente recomendável usar sempre atributos em letras minúsculas e citar sempre valores de atributos para tipos de documentos mais rígidos como XHTML.

Cabeçalhos HTML

Cabeçalhos HTML são títulos ou legendas que se deseja exibir numa página da web.

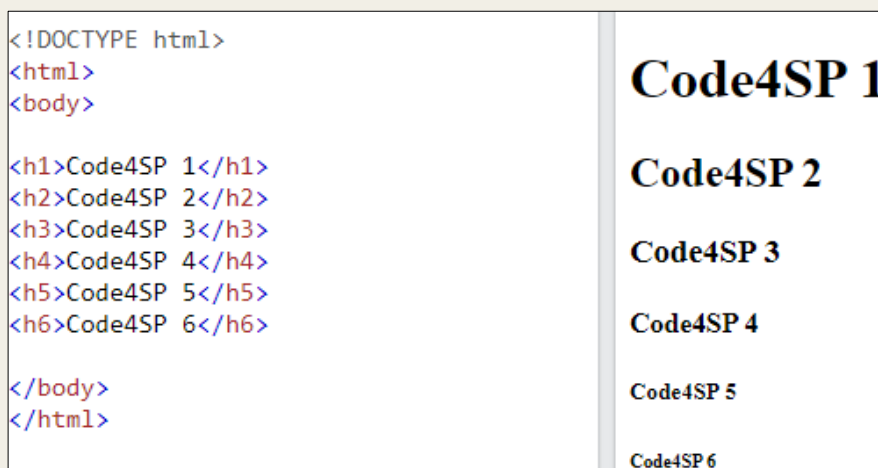


Figure 16 – Código para adicionar cabeçalhos (Fonte: Autor)

Os cabeçalhos HTML são delineados com as tags `<h1>` a `<h6>`. `<h1>` identifica o título mais importante. `<h6>` descreve o título menos importante. Os cabeçalhos `<h1>` devem ser usados para os cabeçalhos principais, seguidos pelos cabeçalhos `<h2>`, depois os menos importantes `<h3>` e assim por diante.

Os títulos são de suma importância, pois os mecanismos de pesquisa os usam para indexar a estrutura e o conteúdo das páginas da web. Os usuários geralmente navegam em uma página por seus títulos. É importante usar títulos para exibir a estrutura do documento.

É importante dizer que, por definição, os navegadores adicionam automaticamente uma margem antes e depois de um título.

Recomendação: É altamente recomendável usar cabeçalhos HTML apenas para cabeçalhos, não para tornar o texto grande ou em negrito.

Além disso, no tópico CSS, será ensinado que o tamanho dos cabeçalhos pode ser especificado usando o atributo `style`, usando a propriedade CSS `font-size`, como segue:

```
<h1 style="font-size:60px;">Heading 1</h1>
```

Parágrafos HTML

Um continua constantemente em uma nova linha e normalmente é um bloco de texto. É definido pelo elemento HTML `<p>` e, como os cabeçalhos, os navegadores adicionam automaticamente margem antes e depois de algum parágrafo.

| | |
|---|--|
| <pre><!DOCTYPE html> <html> <body> <p>Code4SP helps me to code.</p> <p>I love Code4SP.</p> <p>Coding is so great!</p> </body> </html></pre> | <p>Code4SP helps me to code.</p> <p>I love Code4SP.</p> <p>Coding is so great!</p> |
|---|--|

Figura 17 – Código para adicionar parágrafos (Fonte: Autor)

Exibição HTML

Não se pode ter certeza de como o HTML será apresentado, pois pode variar de ecrã para ecrã. Com HTML, a exibição não pode ser alterada adicionando espaços extras ou linhas extras no código HTML.

O navegador removerá automaticamente quaisquer espaços e linhas extras quando a página for exibida, conforme mostrado no exemplo a seguir:

| | |
|---|--|
| <pre><!DOCTYPE html> <html> <body> <p> This paragraph contains a lot of lines in the source code, but the browser ignores it. </p> <p> This paragraph contains a lot of spaces in the source code, but the browser ignores it. </p> <p> The number of lines in a paragraph depends on the size of the browser window. If you resize the browser window, the number of lines in this paragraph will change. </p> </body> </html></pre> | <p>This paragraph contains a lot of lines in the source code, but the browser ignores it.</p> <p>This paragraph contains a lot of spaces in the source code, but the browser ignores it.</p> <p>The number of lines in a paragraph depends on the size of the browser window. If you resize the browser window, the number of lines in this paragraph will change.</p> |
|---|--|

Figura 18 – Exemplo de como os navegadores por vezes ignoram os espaços (Fonte: <https://www.w3schools.com/html>)

Cortes de linha HTML

O elemento HTML `
` define o corte de linha. É uma tag vazia, o que significa que não tem tag de fecho.

`
` deve ser usado se alguém quiser uma nova linha sem começar um novo parágrafo:

| | |
|---|--|
| <pre><!DOCTYPE html> <html> <body> <p>Code4SP really is an amazing project.</p> </body> </html></pre> | <p>Code4SP really is an amazing project.</p> |
|---|--|

Figura 19 – O elemento `
` (Fonte: Autor)

Estilos HTML

O atributo de estilo HTML é usado para adicionar estilos a um elemento, como cor, fonte, tamanho, etc. Para definir o estilo de um elemento HTML, deve-se usar o atributo de estilo. Ele tem a seguinte sintaxe (deve-se notar que propriedade e valor são recursos CSS, a serem aprendidos posteriormente).

```
<tagname style="property:value;">
```

- **Cor de fundo**

A propriedade CSS *background-color* especifica a cor de fundo para um elemento HTML.

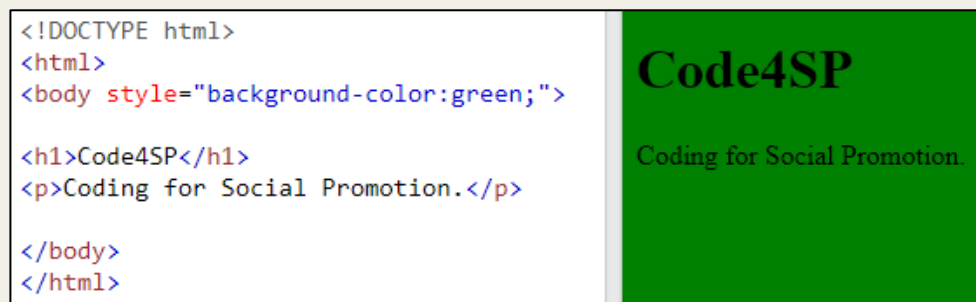


Figura 20 – Definir cor de fundo (Fonte: Autor)

- **Cor de texto**

A propriedade de cor CSS descreve a cor do texto para um elemento HTML:

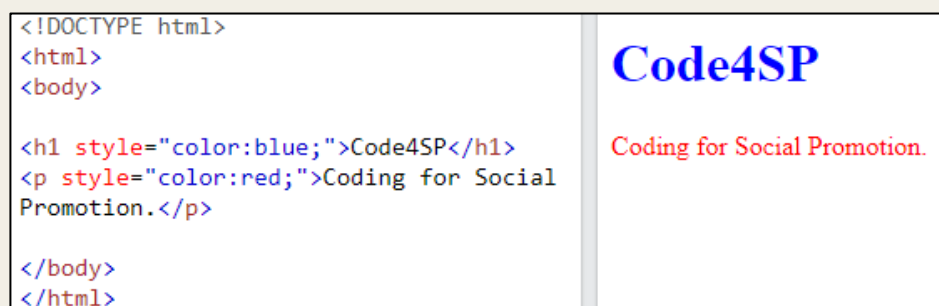


Figura 21 – Definir cor de texto (Fonte: Autor)

- **Fonte**

A propriedade CSS *font-family* define a fonte a ser usada para um elemento HTML:

| | |
|--|--|
| <pre><!DOCTYPE html> <html> <body> <h1 style="font- family:verdana;">Code4SP</h1> <p style="font-family:courier;">Coding for Social Promotion.</p> </body> </html></pre> | <h1>Code4SP</h1> <p>Coding for Social Promotion.</p> |
|--|--|

Figura 22 – Definir a fonte a ser usada para um elemento HTML (Fonte: Autor)

- **Tamanho de texto**

A propriedade CSS *font-size* identifica o tamanho do texto para um elemento HTML:

| | |
|--|--|
| <pre><!DOCTYPE html> <html> <body> <h1 style="font- size:300%;">CODE4SP</h1> <p style="font-size:160%;">Coding for Social Promotion.</p> </body> </html></pre> | <h1>CODE4SP</h1> <p>Coding for Social Promotion.</p> |
|--|--|

Figura 23 – Definir o tamanho de texto para um elemento HTML (Fonte: Autor)

- **Alinhamento de texto**

O recurso de alinhamento de texto CSS define o alinhamento de texto horizontal para um elemento HTML:

| | |
|---|--|
| <pre> <!DOCTYPE html> <html> <body> <h1 style="text-align:center;">CODE4SP</h1> <p style="text-align:center;">Coding for Social Promotion.</p> </body> </html> </pre> | <h1>CODE4SP</h1> <p>Coding for Social Promotion.</p> |
|---|--|

Figura 24 – Característica Alinhamento de texto (Fonte: Autor)

Formatação de texto HTML

HTML compreende vários elementos para definir o texto com uma implicação especial (negrito, itálico, subscrito, sobrescrito, etc.).

Os seguintes são os **elementos de formatação HTML**:

| Propriedade | Resultado | Definição | Exemplo |
|-----------------------------|------------------|--|-------------------------|
| <code></code> | Texto em negrito | O elemento HTML <code></code> especifica texto em negrito, sem nenhuma importância extra. | Texto em negrito |
| <code></code> | Texto importante | O elemento HTML <code></code> descreve o texto com grande importância. O conteúdo interno geralmente é exibido em negrito. | Texto importante |
| <code><i></code> | Texto itálico | O elemento HTML <code><i></code> define uma parte do texto em uma voz ou humor alternativo. O conteúdo | <i>Texto itálico</i> |

| | | | |
|----------------------------|------------------|--|-------------------------------|
| | | dentro é normalmente exibido em itálico. | |
| <code></code> | Texto enfatizado | O elemento HTML <code></code> define o texto enfatizado. O conteúdo dentro é normalmente exibido em itálico. | <i>Texto enfatizado</i> |
| <code><mark></code> | Texto marcado | O elemento HTML <code><mark></code> define o texto que deve ser marcado ou destacado. | Texto marcado |
| <code><small></code> | Texto pequeno | O elemento HTML <code><small></code> define um texto menor. | Texto pequeno |
| <code></code> | Texto apagado | O elemento HTML <code></code> define o texto que foi excluído de um documento. Os navegadores geralmente riscam uma linha através do texto excluído. | Texto apagado |
| <code><ins></code> | Texto inserido | O elemento HTML <code><ins></code> define um texto que foi inserido em um documento. Os navegadores geralmente sublinham o texto inserido: | <u>Texto Inserido</u> |
| <code><sub></code> | Texto subscripto | O elemento HTML <code><sub></code> expressa o texto subscripto. O texto subscripto aparece meio caractere abaixo da linha normal e às vezes é renderizado em uma fonte | Texto Su _b scripto |

| | | | |
|--------------------------|-------------------|---|--------------------------------|
| | | menor. O texto subscrito pode ser usado para Química, como H ₂ O. | |
| <code><sup></code> | Texto sobrescrito | O elemento HTML <code><sup></code> especifica o texto sobrescrito. O texto sobrescrito aparece meio caractere acima da linha normal e às vezes é renderizado em uma fonte menor. O texto sobrescrito pode ser usado para notas de rodapé, como WWW [1]: | Texto su ^P erscrito |

Formatação de texto HTML

HTML `<blockquote>` para Cotações

O elemento HTML `<blockquote>` identifica uma secção que é citada de outra fonte. Os navegadores normalmente recuam elementos `<blockquote>`, como pode ser visto abaixo:

| | |
|---|--|
| <pre><!DOCTYPE html> <html> <body> <p>Browsers typically indent blockquote elements.</p> <blockquote cite="https://code4sp.eu/the-project/"> Code4SP's main objectives and priorities are in full interweaving with the European Commission's goals, contributing towards providing tailored education and training to digitally excluded groups, including migrants and young people from disadvantaged backgrounds, while in parallel, taking into consideration the labor market needs. </blockquote> </body> </html></pre> | <p>Browsers typically indent blockquote elements.</p> <p>Code4SP's main objectives and priorities are in full interweaving with the European Commission's goals, contributing towards providing tailored education and training to digitally excluded groups, including migrants and young people from disadvantaged backgrounds, while in parallel, taking into consideration the labor market needs.</p> |
|---|--|

Figura 25 – O elemento Blockquote (Fonte: Autor)

HTML `<q>` para pequenas Cotações

A tag HTML `<q>` especifica uma citação curta. Os navegadores geralmente inserem aspas ao redor da cotação, como segue:

| | |
|---|--|
| <pre><!DOCTYPE html> <html> <body> <p>Browsers usually insert quotation marks around the q element.</p> <p>WWF's goal is to: <q>Build a future where people live in harmony with nature.</q></p> </body> </html></pre> | <p>Browsers usually insert quotation marks around the q element.</p> <p>WWF's goal is to: "Build a future where people live in harmony with nature."</p> |
|---|--|

Figura 26 – O elemento de pequenas citações (Fonte: https://www.w3schools.com/html/html_quotation_elements.asp)

HTML `<abbr>` para abreviações

A tag HTML `<abbr>` especifica uma abreviação ou um acrônimo, como "HTML", "CSS", "Mr.", "Dr.", "ASAP", etc. Abreviações de marcação podem fornecer informações valiosas para navegadores, sistemas de tradução e motores de busca, como visto anteriormente. Caso não se conheça o significado de alguma abreviatura, pode-se usar o atributo global `title` para mostrar a descrição da abreviatura/sigla ao passar o rato sobre o elemento, como visto abaixo:


| | |
|--|---|
| <pre><!DOCTYPE html> <html> <body> <p>The <abbr title="World Health Organization">WHO</abbr> was founded in 1948.</p> <p>Marking up abbreviations can give useful information to browsers, translation systems and search-engines. </p> </body> </html></pre> | <p>The WHO was founded in 1948.</p> <p>Marking up World Health Organization useful information to browsers, translation systems and search-engines.</p>  |
|--|---|

Figura 27 – O elemento `<abbr>` e a função mouse over (Fonte: Autor)

HTML <address> para Informação de Contacto

A tag HTML <address> define as informações de contato do autor/proprietário de um documento ou artigo. Pode ser um endereço de e-mail, URL, endereço físico, número de telefone, etc. O texto contido no elemento <address> normalmente é apresentado em itálico e os navegadores adicionarão sempre uma quebra de linha antes e depois dele, como segue:

| | |
|---|---|
| <pre><!DOCTYPE html> <html> <body> <p>Please contact us:</p> <address> https://code4sp.eu/ https://www.facebook.com/Code4SP </address> </body> </html></pre> | <p>Please contact us:</p> <p><i>https://code4sp.eu/</i> <i>https://www.facebook.com/Code4SP</i></p> |
|---|---|

Figura 28 – O elemento <address> (Fonte: Autor)

HTML <cite> para Título de Trabalho

A tag HTML <cite> define o título de um livro, um poema, uma música, um filme, uma pintura e todos os trabalhos criativos. Deve-se afirmar que o nome do autor não é o título de uma obra.

Como nas tags acima, o texto no elemento <cite> normalmente é renderizado em itálico:



Figura 29 – O elemento <cite> (Fonte: https://www.w3schools.com/html/html_quotation_elements.asp)

HTML <bdo> for Substituição Bidirecional

A tag HTML <bdo> significa "substituição bidirecional", que é usada para substituir a direção do texto atual/padrão. Essa tag define a direção do conteúdo dentro dela para executar no navegador da esquerda para a direita ou da direita para a esquerda (rtl – direita para esquerda; ltr – esquerda para direita).

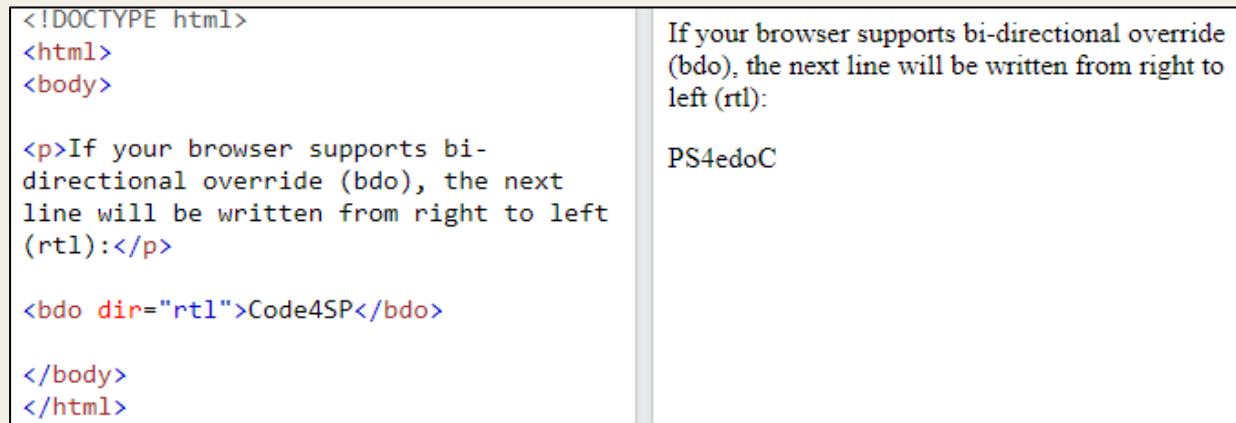


Figura 30 – O elemento <bdo> (Fonte: https://www.w3schools.com/html/html_quotation_elements.asp)

Comentários HTML

Adicionar Comentários

Este elemento é usado para adicionar um comentário a um documento HTML. Um comentário HTML começa com `<!--` e termina com `-->`. Os comentários HTML são visíveis para qualquer pessoa que visualize o código-fonte da página, mas não são renderizados quando o documento HTML é renderizado por um navegador. Deve-se notar que há um ponto de exclamação na tag inicial, mas não na tag final. Este recurso é especialmente útil para colocar notificações e lembretes no código HTML:

```
<!DOCTYPE html>
<html>
<body>

<!-- This is a comment -->
<p>Code4SP project.</p>
<!-- Comments are not displayed in the
browser -->

</body>
</html>
```

Code4SP project.

Figura 31 – A funcionalidade dos comentários (Fonte: Autor)

Esconder conteúdo

Os comentários também podem ser usados para ocultar o conteúdo, e isso pode ser útil se alguém o ocultar no momento. Pode também ocultar mais do que uma linha, tudo entre `<!--` e `-->` será ocultado do ecrã. Os comentários também são ótimos para depurar HTML, porque é possível comentar linhas de código HTML, uma de cada vez, para procurar erros.

| | |
|---|--|
| <pre> <!DOCTYPE html> <html> <body> <p>Code4SP project.</p> <!-- <p>This content is hidden. </p> -- > <p>But this will appear.</p> </body> </html> </pre> | <p>Code4SP project.</p> <p>But this will appear.</p> |
|---|--|

Figure 33 – A funcionalidade de esconder conteúdo (Fonte: Autor)

Cores HTML

As cores HTML são estipuladas com nomes predefinidos de cores ou com RGB, HEX, HSL, RGBA, ou valores HSLA.

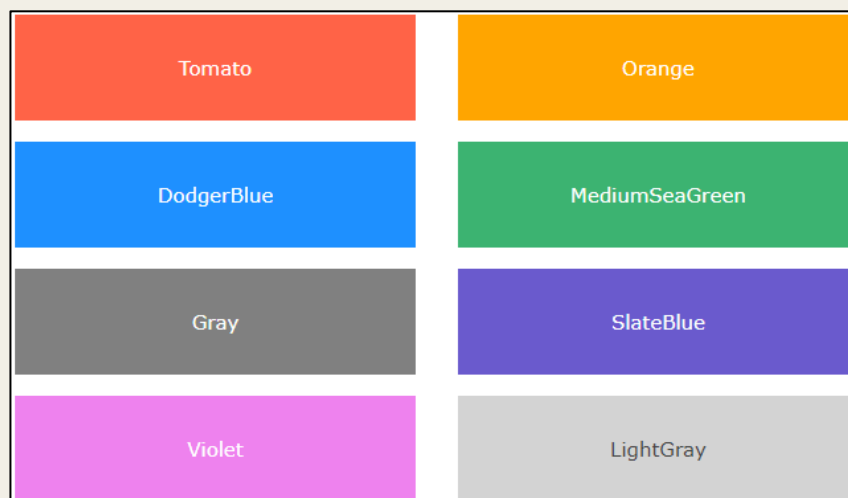


Figure 34 – Alguns nomes de cores que se podem definir. (Fonte: https://www.w3schools.com/html/html_colors.asp)

As cores podem ser definidas para **o fundo de página**:


| | |
|---|--|
| <pre><!DOCTYPE html> <html> <body> <h1 style="background-color: DodgerBlue;">Code4SP</h1> <p style="background-color:Tomato;"> Code4SP's main objectives and priorities are in full interweaving with the European Commission's goals, contributing towards providing tailored education and training to digitally excluded groups, including migrants and young people from disadvantaged backgrounds, while in parallel, taking into consideration the labor market needs. </p> </body> </html></pre> |  |
|---|--|

Figura 34 – Definir a cor de fundo de uma Webpage. (Fonte: Autor)

O mesmo princípio pode ser aplicado à cor de texto:

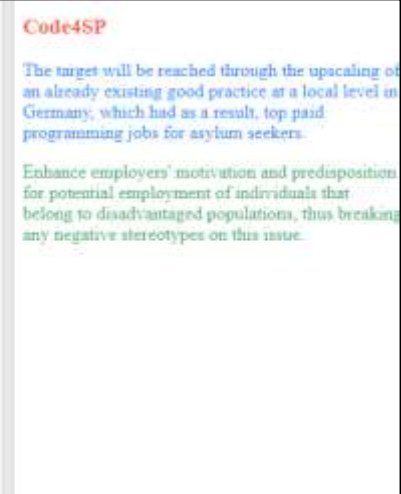
| | |
|--|---|
| <pre><!DOCTYPE html> <html> <body> <h3 style="color:Tomato;">Code4SP</h3> <p style="color:DodgerBlue;">The target will be reached through the upscaling of an already existing good practice at a local level in Germany, which had as a result, top paid programming jobs for asylum seekers.</p> <p style="color:MediumSeaGreen;">Enhance employers' motivation and predisposition for potential employment of individuals that belong to disadvantaged populations, thus breaking any negative stereotypes on this issue.</p> </body> </html></pre> |  |
|--|---|

Figura 35 – Definir a cor de texto (Fonte: Autor)

Também para as cores da **borda**:

| | |
|---|---|
| <pre><!DOCTYPE html> <html> <body> <h1 style="border: 2px solid Tomato;">Code4SP</h1> <h1 style="border: 2px solid DodgerBlue;">Code4SP</h1> <h1 style="border: 2px solid Violet;">Code4SP</h1> </body> </html></pre> | <div style="border: 2px solid red; padding: 5px; margin-bottom: 10px;">Code4SP</div> <div style="border: 2px solid blue; padding: 5px; margin-bottom: 10px;">Code4SP</div> <div style="border: 2px solid purple; padding: 5px;">Code4SP</div> |
|---|---|

Figura 36 – Adicionar bordas e definir a sua cor (Fonte: Autor)

Valores de cor

Conforme mencionado acima, as cores HTML são estipuladas com nomes de cores predefinidos ou com valores RGB, HEX, HSL, RGBA ou HSLA. Os três elementos `<div>` a seguir têm sua cor de fundo definida com valores RGB, HEX e HSL:

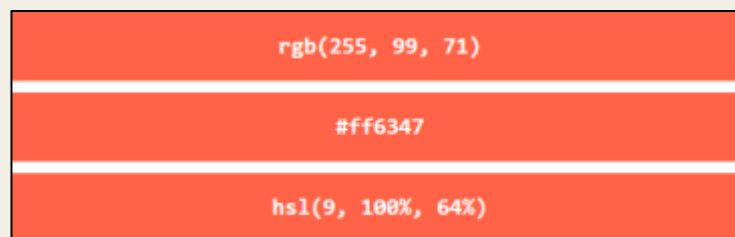


Figura 37 – Valores RGB, HEX e HSL para a cor Tomato (Fonte: https://www.w3schools.com/html/html_colors.asp)

A transparência também é um recurso que pode ser adicionado ao definir uma cor, adicionando-lhe um canal Alfa. O exemplo a seguir como 50% de transparência:

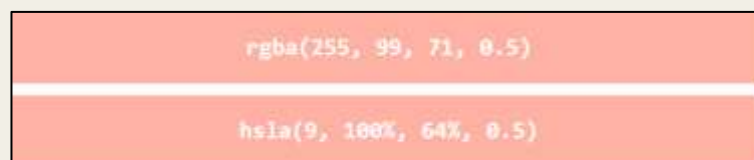


Figura 38 – Definir valores de transparência para a cor Tomato (Fonte: https://www.w3schools.com/html/html_colors.asp)

O código para configurar os dois recursos facilitará a compreensão dos alunos. Como visto, ele compreende recursos CSS a serem explorados posteriormente:

| | |
|---|---|
| <pre><!DOCTYPE html> <html> <body> <p>Same as color name "Tomato":</p> <h1 style="background-color:rgb(255, 99, 71);">rgb(255, 99, 71)</h1> <h1 style="background- color:#ff6347;">#ff6347</h1> <h1 style="background-color:hsl(9, 100%, 64%);">hsl(9, 100%, 64%)</h1> <p>Same as color name "Tomato", but 50% transparent:</p> <h1 style="background-color:rgba(255, 99, 71, 0.5);">rgba(255, 99, 71, 0.5) </h1> <h1 style="background-color:hsla(9, 100%, 64%, 0.5);">hsla(9, 100%, 64%, 0.5)</h1> <p>In addition to the predefined color names, colors can be specified using RGB, HEX, HSL, or even transparent colors using RGBA or HSLA color values. </p> </body> </html></pre> | <p>Same as color name "Tomato":</p> <p>rgb(255, 99, 71)</p> <p>#ff6347</p> <p>hsl(9, 100%, 64%)</p> <p>Same as color name "Tomato", but 50% transparent:</p> <p>rgba(255, 99, 71, 0.5)</p> <p>hsla(9, 100%, 64%, 0.5)</p> <p>In addition to the predefined color names, colors can be specified using RGB, HEX, HSL, or even transparent colors using RGBA or HSLA color values.</p> |
|---|---|

Figura 39 – Definir cor e transparência da cor Tomato (Fonte: https://www.w3schools.com/html/html_colors.asp)

Mais informações em relação a valores RGB, HEX, HSL, RGBA ou HSLA pode ser encontrada em: https://www.w3schools.com/html/html_colors.asp

Ligações HTML

As ligações podem ser encontradas em quase todas as páginas da web. Elas permitem que os utilizadores da Internet naveguem de página em página. Não precisa ser texto, pode ser uma imagem ou qualquer outro elemento HTML.

Hyperlinks

Hyperlinks HTML são ligações que podem ser clicadas, direcionando para outro documento.

Quando o rato é movido sobre uma ligação, a seta do rato transforma-se numa pequena mão.

Sintaxe

A tag HTML `<a>` define um hyperlink. Tem a seguinte sintaxe:

```
<a href="url">link text</a>
```

O atributo mais significativo do elemento `<a>` é o atributo `href`, que indica o destino do link. O texto do link é a parte que ficará visível para o utilizador. Ao clicar no texto do link, o leitor será redirecionado para o endereço URL necessário.

Por padrão, os links serão vistos em todos os navegadores da seguinte forma:

- Um link não visitado é **sublinhado e azul**
- Um link visitado é **sublinhado e roxo**
- Um link ativo é **sublinhado e vermelho**

Nota: As cores dos links podem ser alteradas ao usar funcionalidades do CSS.

O atributo de destino

Por padrão, a página vinculada será mostrada na janela atual do navegador. Para modificar isso, os alunos devem indicar outro destino para o link.

O **atributo** de destino indica onde abrir o documento vinculado. Pode ter um dos seguintes valores:

- **_self** - Padrão. Abre o documento na mesma janela/guia em que foi clicado
- **_blank** - Abre o documento numa nova janela ou guia
- **_parent** - Abre o documento no parent frame
- **_top** - Abre o documento em todo o corpo da janela

Usar imagem como ligação

Para usar uma imagem como link, basta colocar a tag **** dentro da tag **<a>**, como pode ser verificado no tutorial a seguir:

https://www.w3schools.com/html/tryit.asp?filename=tryhtml_links_image

Ligar um endereço de email

Para criar uma ligação que abra o software de e-mail do utilizador (permitindo que ele envie um novo e-mail), **mailto:** deve ser adicionado dentro do atributo **href**, seguindo o próximo exemplo:

```
<a href="mailto:someone@example.com">Send email</a>
```

Criar um marcador em HTML

Ligações HTML podem ser aplicadas para criar marcadores, para que os leitores possam ir para partes específicas de uma página da Web, podendo ser realmente útil se a página Web for muito longa. Este processo é composto por duas etapas muito simples:

- Para criar um marcador - primeiro o marcador deve ser criado e, em seguida, uma ligação deve ser-lhe adicionada. Para criar, deve-se utilizar o atributo **id** (ex.: `<h2 id="C1">Capítulo 1</h2>`), em seguida, deve-se adicionar uma ligação para o favorito, dentro da mesma página (ex.: `Pular para o Capítulo 4`)
- Quando a ligação for clicada, a página irá para o local com o marcador.

Imagens HTML

Inserir imagens em páginas Web

As imagens melhoram a aparência visual das páginas web, tornando-as mais atraentes e coloridas. A tag `` é usada para adicionar imagens nas páginas HTML. É um elemento vazio e contém apenas atributos.

Cada imagem deve ter pelo menos dois atributos: os atributos **src** e **alt**. O atributo **src** informa o navegador onde encontrar a imagem, sendo o seu valor o URL do arquivo de imagem. O atributo **alt** fornece um texto alternativo para a imagem se ela estiver inacessível ou não puder ser exibida por algum motivo (conexão lenta, imagem não disponível na URL especificada ou se o utilizador usar um leitor de ecrã ou navegador não gráfico). O seu valor deve ser um substituto significativo para a imagem, preferencialmente um texto sugestivo.



Figura 40 – Adicionar uma imagem a um documento HTML, usando os atributos **src** e **alt** (Fonte: Autor)

Definir a Largura e Altura de uma Imagem

Os atributos **largura** e **altura** são usados para indicar a largura e a altura de uma imagem. Os valores desses atributos são interpretados em pixels por padrão. É uma boa prática especificar os atributos de largura e altura, para que o navegador possa atribuir o espaço necessário para a imagem, antes de esta ser transferida.



Figura 41 – Definir a altura e a largura de uma imagem (Fonte::

<https://www.tutorialrepublic.com/codelab.php?topic=html&file=specify-dimensions-for-images>)

O atributo **estilo** também pode ser usado para indicar largura e altura. Impede que as folhas de estilo alterem o tamanho da imagem acidentalmente, porque o estilo embutido tem a prioridade mais elevada.

Usar o elemento de imagem HTML5

De vez em quando, dimensionar uma imagem para cima ou para baixo para caber em diferentes dispositivos (ou tamanhos de ecrã) não funciona conforme o esperado. Além disso, reduzir a dimensão da imagem usando o atributo **largura** e **altura** não diminui o tamanho inicial do arquivo. Para resolver esses problemas, o HTML5

introduziu a tag `<picture>`, que permite definir várias versões de uma imagem para atingir 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 `` no final. Da mesma forma, cada elemento `<source>` tem o atributo `media` que especifica uma condição de media que é usada pelo navegador para determinar quando uma determinada fonte deve ser usada.

Mapa de Imagem

Um mapa de imagem permite definir pontos de acesso numa imagem que funciona como um hyperlink. A ideia-chave por trás da criação de um mapa de imagem é fornecer uma maneira simples de vincular várias partes de uma imagem sem dividi-la em arquivos de imagem separados. Por exemplo, um mapa de um país pode ter cada cidade vinculada a mais informações sobre essa cidade.

O exemplo a seguir é bastante preciso sobre esses recursos:

<https://www.tutorialrepublic.com/codelab.php?topic=html&file=image-maps>

O atributo `name` da tag `<map>` é usado para referenciar o mapa da tag `` usando seu atributo `usemap`. A tag `<area>` é usada dentro do elemento `<map>` para definir as áreas clicáveis numa imagem. Qualquer número de áreas clicáveis pode ser definido numa imagem.

Favicon HTML

Um favicon é uma pequena imagem exibida à esquerda do título da página na guia do navegador:

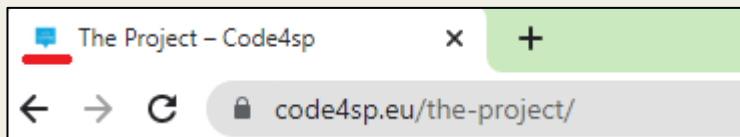


Figura 42 – Favicon do Code4SP's (Fonte: Autor)

Para adicionar um favicon a um site, uma imagem favicon deve ser guardada na diretória raiz do servidor web. Outra forma é criar uma pasta na diretória raiz chamada imagens e guardar a imagem do favicon nessa pasta. Um nome comum para uma imagem favicon é "favicon.ico".

Em seguida, um elemento `<link>` deve ser adicionado ao arquivo "index.html", após o elemento `<title>`, conforme segue:

```
<!DOCTYPE html>
<html>
<head>
  <title>Code4SP</title>
  <link rel="icon" type="image/x-icon" href="/images/favicon.ico">
</head>
<body>
<h1>Our project</h1>
<p>Co-funded by the E+ fund of the EC.</p>
</body>
</html>
```

Tabelas HTML

Criar tabelas em HTML

As tabelas HTML permitem organizar os dados em linhas e colunas. Eles geralmente são usados para exibir dados tabulares, como listagens de produtos, detalhes do cliente, relatórios financeiros, etc.

Uma tabela pode ser criada usando o elemento `<table>`. Dentro do elemento `<table>`, os elementos `<tr>` podem ser utilizados para criar linhas, e para criar colunas dentro de uma linha, os elementos `<td>` podem ser usados. Uma célula pode ser definida como um cabeçalho para um grupo de células da tabela usando o elemento `<th>`.

As tabelas não têm bordas por padrão. A propriedade CSS `border` pode ser usada para adicionar bordas às tabelas. Além disso, as células da tabela são dimensionadas apenas o suficiente para caber o conteúdo por padrão. Para adicionar mais espaço ao redor do conteúdo nas células da tabela, a propriedade de `preenchimento` CSS pode ser usada.

Por padrão, as bordas ao redor da tabela e suas células são separadas umas das outras. Mas eles podem ser reduzidos numa só, 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, a propriedade CSS `text-align` pode ser usada. Para isso, sugere-se que os alunos atinjam primeiro o tópico CSS. A maioria dos atributos do elemento `<table>`, como `border`, `cellpadding`, `cellspacing`, `width`, `align`, etc. para estilizar aparências de tabelas em versões anteriores foram descartados no HTML5, portanto, devem ser evitados. CSS deve ser **privilegiado para estilizar tabelas HTML**.

```

<!DOCTYPE html>
<html lang="en">
<head>
  <title>Creating Tables in HTML</title>
</head>
<body>
  <h2>Spotify Top Songs of 2021 (USA)</h2>
  <table>
    <tr>
      <th>No.</th>
      <th>Song - Band </th>
      <th>Lenght</th>
    </tr>
    <tr>
      <td>1</td>
      <td>drivers licence - Olivia Rodrigo</td>
      <td>4:02</td>
    </tr>
    <tr>
      <td>2</td>
      <td>MONTERO (Call me by your name) - Lil Nas X</td>
      <td>2:17</td>
    </tr>
    <tr>
      <td>3</td>
      <td>STAY - The Kid LAROI ft. Justin Bieber</td>
      <td>2:21</td>
    </tr>
  </table>
</body>
</html>

```

Spotify Top Songs of 2021 (USA)

| No. | Song - Band | Lenght |
|-----|--|--------|
| 1 | drivers licence - Olivia Rodrigo | 4:02 |
| 2 | MONTERO (Call me by your name) - Lil Nas X | 2:17 |
| 3 | STAY - The Kid LAROI ft. Justin Bieber | 2:21 |

Figura 43 – Uma tabela básica (Fonte: Autor)

Abranger várias linhas e colunas

A abrangência permite estender 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. No entanto, os atributos **rowspan** ou **colspan** podem ser usados para abranger várias linhas ou colunas numa tabela.

Da mesma forma, o atributo **rowspan** pode ser usado para criar uma célula que abrange mais de uma linha, como segue:

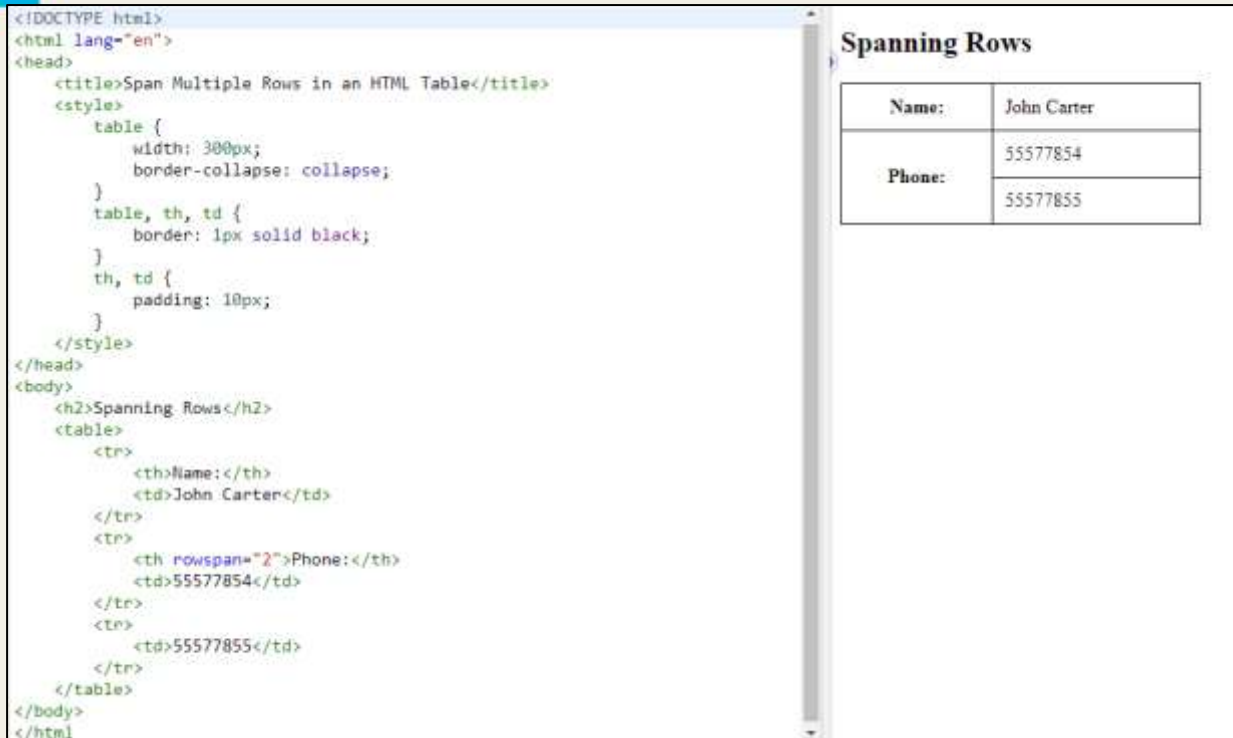


Figure 45 – O atributo rowspan (Fonte: <https://www.tutorialrepublic.com/html-tutorial/html-tables.php>)

Legendas da tabela

Uma legenda (ou título) para tabelas pode ser criada usando o elemento <caption>. Este elemento deve ser colocado diretamente após a tag (de abertura) <table>. Por padrão, a legenda aparece na parte superior da tabela, mas isso pode ser alterado usando a propriedade CSS caption-side.



Figure 45 – Adicionar legendas de tabela (Fonte: <https://www.wikitechy.com>)

Definir um cabeçalho, corpo e rodapé de tabela

HTML fornece uma série de tags `<thead>`, `<tbody>` e `<tfoot>` que ajudam os alunos a criar tabelas mais coordenadas, definindo regiões de cabeçalho, corpo e rodapé, nessa ordem.

```

<!DOCTYPE html>
<html lang="en">
<head>
  <title>HTML Table with a Header, Footer and Body</title>
  <style>
    table {
      width: 300px;
      border-collapse: collapse;
    }
    table, th, td {
      border: 1px solid black;
    }
    th, td {
      padding: 10px;
      text-align: left;
    }
  </style>
</head>
<body>
  <table>
    <thead>
      <tr>
        <th>Items</th>
        <th>Expenditure</th>
      </tr>
    </thead>
    <tbody>
      <tr>
        <td>Stationary</td>
        <td>2,000</td>
      </tr>
      <tr>
        <td>Furniture</td>
        <td>10,000</td>
      </tr>
    </tbody>
    <tfoot>
      <tr>
        <th>Total</th>
        <td>12,000</td>
      </tr>
    </tfoot>
  </table>
</body>
</html>

```

| Items | Expenditure |
|--------------|-------------|
| Stationary | 2,000 |
| Furniture | 10,000 |
| Total | 12,000 |

Figura 45 – Adicionar legendas de tabela (Fonte: <https://www.tutorialrepublic.com/html-tutorial/html-tables.php>)

Listas HTML

As listas HTML são aplicadas para apresentar informações de forma bem formada. Dentro delas, pode-se adicionar texto, imagens, ligações, quebras de linha, etc. Existem três tipos diferentes de listas em HTML e cada uma tem um propósito e significado específico:

- **A) Listas não ordenadas** — Usadas para criar uma lista de itens relacionados, sem ordem específica.
- **B) Listas ordenadas** — Usadas para criar uma lista de itens relacionados, em uma determinada ordem.
- **C) Listas de descrição** — Usadas para criar uma lista de termos e suas descrições.

A) Listas não ordenadas

Uma lista não ordenada é criada usando o elemento ``, e cada item da lista começa com o elemento ``. Está marcado com pontos, como segue:



```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Reasons why you should travel by train</title>
</head>
<body>
  <h2>Reasons why you should travel by train</h2>
  <ul>
    <li>It is less expensive</li>
    <li>It is eco-friendly</li>
    <li>You can enjoy the view</li>
  </ul>
</body>
</html>
```

Reasons why you should travel by train

- It is less expensive
- It is eco-friendly
- You can enjoy the view

Figura 46 – Listas não ordenadas (Fonte: Autor)

A) Listas ordenadas

Uma lista ordenada é criada usando o elemento `` e cada item da lista começa com o elemento ``. As listas ordenadas são usadas quando a ordem dos itens da lista é crítica. É marcado com números, como segue:

| | |
|---|---|
| <pre> <!DOCTYPE html> <html lang="en"> <head> <title>HTML Ordered list</title> </head> <body> <h2>How to cook your own veggie burger</h2> Dump your ground meat into a bowl. (We go for ground meat with around 20% fat.) Season it with salt, pepper, and whatever else you want; you can add spices, perhaps, or Worcestershire sauce, or shallots, or chiles. Shape your burgers into patties, using your thumb to make an indentation in the center; this will keep the burgers from puffing up. Keep in mind that the burgers will shrink up a bit once you cook them, so make your patties a bit bigger than you want them later. Oil your grill or a cast-iron pan, and grill or sear those patties. (How many times to flip them is up for debate -- but when I'm grilling, I flip once so I can get those nice grill marks.) Cook them until your desired doneness (around 125-130°F for medium rare, around 1 minute per side for each inch of thickness). But before you take them off the grill... ...add your cheese and toast your buns. Let the cheese melt while the burgers are still on the grill; to speed things up, you can close the cover. Once your burgers are finished cooking, and your cheese is melty and your buns are nicely charred, throw some condiments and toppings on those burgers. Anything goes. (Really, anything goes.) Bite into it and let those juices run down your chin, and rejoice that it's summer. And then make another round, because now you know how. </body> </html> </pre> | <h3>How to cook your own veggie burger</h3> <ol style="list-style-type: none"> 1. Dump your ground meat into a bowl. (We go for ground meat with around 20% fat.) Season it with salt, pepper, and whatever else you want; you can add spices, perhaps, or Worcestershire sauce, or shallots, or chiles. 2. Shape your burgers into patties, using your thumb to make an indentation in the center; this will keep the burgers from puffing up. Keep in mind that the burgers will shrink up a bit once you cook them, so make your patties a bit bigger than you want them later. 3. Oil your grill or a cast-iron pan, and grill or sear those patties. (How many times to flip them is up for debate -- but when I'm grilling, I flip once so I can get those nice grill marks.) Cook them until your desired doneness (around 125-130°F for medium rare, around 1 minute per side for each inch of thickness). But before you take them off the grill... 4. ...add your cheese and toast your buns. Let the cheese melt while the burgers are still on the grill; to speed things up, you can close the cover. 5. Once your burgers are finished cooking, and your cheese is melty and your buns are nicely charred, throw some condiments and toppings on those burgers. Anything goes. (Really, anything goes.) Bite into it and let those juices run down your chin, and rejoice that it's summer. And then make another round, because now you know how. |
|---|---|

Figura 47 – Listas ordenadas (Fonte: Autor)

B) Listas de descrição

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 renderizam as listas de definições colocando os termos e as definições em linhas separadas, onde as definições dos termos são levemente recuadas.

| | |
|--|--|
| <pre> <!DOCTYPE html> <html lang="en"> <head> <title>HTML Description or Definition List</title> </head> <body> <h2>What are bread and coffee?</h2> <dl> <dt>Bread</dt> <dd>A baked food made of flour.</dd> <dt>Coffee</dt> <dd>A drink made from roasted coffee beans.</dd> </dl> </body> </html> </pre> | <h3>What are bread and coffee?</h3> <p>Bread A baked food made of flour.</p> <p>Coffee A drink made from roasted coffee beans.</p> |
|--|--|

Figura 48 – Listas de descrição (Fonte: Autor)

Formulários HTML

Espera-se que os formulários HTML colem diferentes tipos de entradas do utilizador, como detalhes de contacto (nome, e-mail, número de telefone, conta bancária, etc.). Os formulários incluem elementos especiais conhecidos como controlos, por exemplo caixa de entrada, caixas de seleção, botões de opção, botões de envio, etc. esses dados.

A tag `<form>` é usada para gerar um formulário HTML. Um exemplo simples de um formulário de login seria o seguinte:



Figura 49 – Exemplo de um formulário de login (Fonte: <https://www.tutorialrepublic.com/html-tutorial/html-forms.php>)

Existem diferentes tipos de controlos a serem aplicados num formulário HTML, sendo os elementos de entrada os mais utilizados. Eles identificam vários tipos de campos de entrada do utilizador, dependendo do atributo de tipo. Os elementos de entrada podem ser campos de texto, campos de senha, caixas de seleção, botões de envio, botões de reinicialização, caixas de seleção de arquivos, bem como vários novos tipos de entrada introduzidos no HTML5 (eles podem ser verificados aqui).

Campos de texto são áreas que permitem aos utilizadores adicionar **texto**. Eles são criados usando um elemento `<input>`, cujo atributo **type** tem um valor de texto. Deve-se notar que a tag `<label>` é usada para identificar os rótulos dos elementos `<input>`.

Se o webmaster quiser que os seus utilizadores insiram várias linhas, `<textarea>` deve ser adicionado.

| | |
|--|------------------------------------|
| <pre><!DOCTYPE html> <html lang="en"> <head> <title>HTML Text Input Field</title> </head> <body> <form> <label for="user-name">Login:</label> <input type="text" name="username" id="user-name"> </form> </body> </html></pre> | <p>Login: <input type="text"/></p> |
|--|------------------------------------|

Figura 50 – Campo de texto (Fonte: Autor)

Os campos de senha são como campos de texto, sendo que os únicos caracteres de diferença em um campo de senha são ocultos, portanto, são exibidos como asteriscos ou pontos. Da mesma forma, esse procedimento impede que outra pessoa leia a senha na tela. Este também é um controle de entrada de texto de linha única gerado usando um elemento `<input>` cujo atributo `type` tem um valor de senha.

| | |
|--|---|
| <pre><!DOCTYPE html> <html lang="en"> <head> <title>HTML Password Input Field</title> </head> <body> <form> <label for="user-pwd">Password:</label> <input type="password" name="user-password" id="user-pwd"> </form> </body> </html></pre> | <p>Password: <input type="password"/></p> |
|--|---|

Figura 51 – Exemplo de campo de password (Fonte: <https://www.tutorialrepublic.com/html-tutorial/html-forms.php>)

Os botões de opção são aplicados para permitir que o utilizador selecione exatamente uma opção de um conjunto pré-especificado de opções. Ele é gerado usando um elemento `<input>` cujo atributo `type` tem um valor de radio.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>HTML Radio Buttons</title>
</head>
<body>
  <form>
    <input type="radio" name="Civil Status" value="single" id="single">
    <label for="single">Single</label>
    <input type="radio" name="Civil Status" value="married" id="married">
    <label for="married">Married</label>
    <input type="radio" name="Civil Status" value="other" id="other">
    <label for="other">Other</label>
  </form>
</body>
</html>
```

Single Married Other

Figura 52 – Código para configurar botões de rádio (Fonte: Autor)

As **caixas de seleção** disponibilizam ao usuário uma ou mais opções de um conjunto predefinido de opções. Ele é gerado usando um elemento `<input>` cujo atributo `type` tem um valor de `checkbox`. Se preferir gerar uma caixa de seleção (ou botão de rádio) selecionada por padrão, basta adicionar o atributo `checked` ao elemento de entrada (`<input type="checkbox" checked>`).

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>HTML Checkboxes</title>
</head>
<body>
  <form>
    <input type="checkbox" name="sports" value="football" id="football">
    <label for="football">Football</label>
    <input type="checkbox" name="sports" value="handball" id="handball">
    <label for="handball">Handball</label>
    <input type="checkbox" name="sports" value="basketball" id="basketball">
    <label for="basketball">Basketball</label>
  </form>
</body>
</html>
```

Football Handball Basketball

Figura 53 – Código para definir checkboxes (Fonte: Autor)

As caixas de seleção de arquivo permitem que um utilizador procure um arquivo local e o envie como um anexo com os dados do formulário. Por exemplo, o Google Chrome fornece um campo de entrada de seleção de arquivo com um botão "Procurar" que permite ao usuário selecionar um arquivo de seu disco rígido.

As caixas de seleção de arquivo também são criadas usando um elemento `<input>`, cujo valor do atributo `type` é definido como `file`.

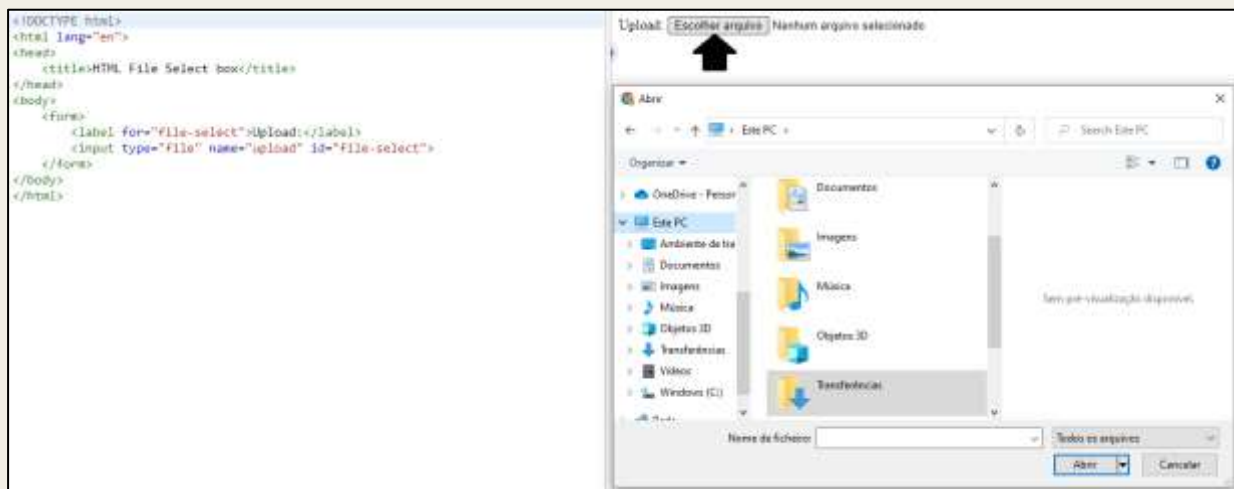


Figura 54 – Código para configurar caixas de seleção de arquivos (Fonte: Autor)

Textarea pode ser definida como um controle de entrada de texto de várias linhas, que permite inserir mais de uma linha de texto. Esses controles são criados usando um elemento `<textarea>`, como segue:



Figura 55 – Código para configurar um controle de entrada de texto de várias linhas (Fonte: Autor)

As caixas de seleção são listas suspensas de opções nas quais um usuário pode selecionar uma ou mais opções num menu suspenso. Eles são criados usando o elemento `<select>` e o elemento `<option>`. Os elementos `<option>` dentro do elemento `<select>` definem cada item da lista.

```

<!DOCTYPE html>
<html lang="en">
<head>
  <title>HTML Select Box</title>
</head>
<body>
  <form>
    <label for="country">Country:</label>
    <select name="country" id="country">
      <option value="Portugal">Portugal</option>
      <option value="Cyprus">Cyprus</option>
      <option value="Greece">Greece</option>
    </select>
  </form>
</body>
</html>

```

Figura 56 – Código para configurar caixas de seleção (Fonte: Autor)

Os botões Enviar e Redefinir são muito comuns na maioria dos sites. Os botões de envio são usados para enviar (formulário) dados para um servidor da Web, enquanto os botões de redefinição são criados para redefinir o formulário para os valores padrão. Quando o usuário clica no botão enviar, os dados do formulário são enviados para o arquivo especificado no atributo de ação do formulário para processar os dados enviados.

Os botões também podem ser criados usando o elemento `<button>`. Eles têm a mesma finalidade dos botões criados com o elemento `input`. No entanto, eles oferecem mais possibilidades de renderização, pois permitem a incorporação de outros [elementos HTML](#).

```

<!DOCTYPE html>
<html lang="en">
<head>
  <title>HTML Submit and Reset Buttons</title>
</head>
<body>
  <form action="/examples/html/action.php" method="post">
    <label for="first-name">First Name:</label>
    <input type="text" name="first-name" id="first-name">
    <input type="submit" value="Submit">
    <input type="reset" value="Reset">
  </form>
</body>
</html>

```

Figura 57 – Código para configurar os botões de envio e redefinição (Fonte: Autor)

Agrupar controles de formulário é uma ótima ferramenta para os usuários localizarem um controle, tornando o formulário mais acessível. O elemento `<legend>` é fundamental para criar controles logicamente relacionados, como visto na figura abaixo:

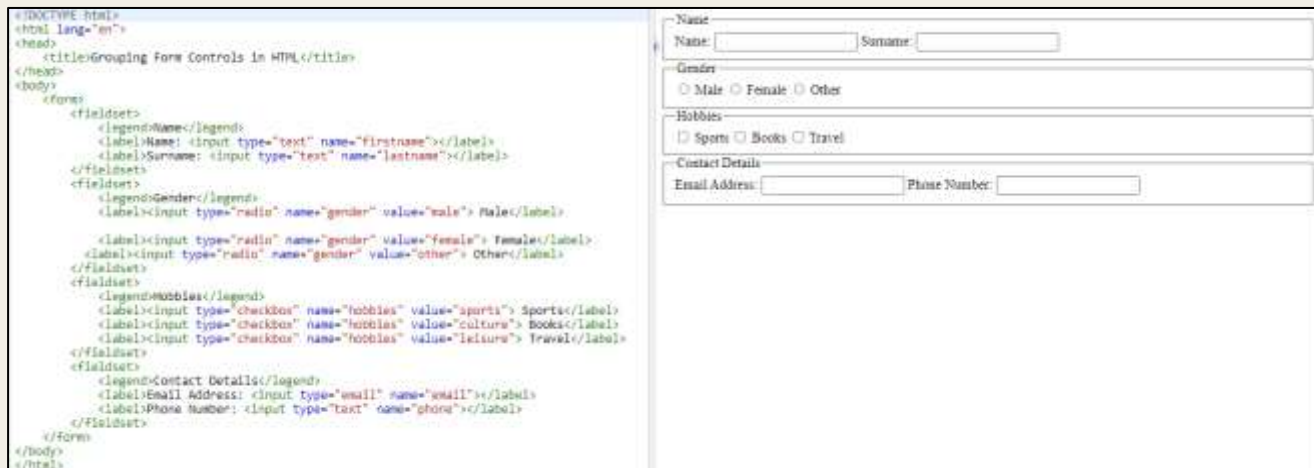


Figura 58 – Código para configurar controles de formulário de agrupamento (Fonte: Autor, check [this](#) for better definition)

Abaixo tem uma lista de atributos de elementos de formulário usados com frequência:

| Atributo | Descrição |
|----------|--|
| name | Especifica o nome do formulário. |
| action | Especifica o URL do programa ou script no servidor web que será usado para processar as informações enviadas via formulário. |
| method | Especifica o método HTTP usado para enviar os dados ao servidor 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, _parent e _top. |
| enctype | Especifica como os dados do formulário devem ser codificados ao enviar o formulário ao servidor. Aplicável somente quando o valor do atributo method for post. |

Tabela 2 – Lista de atributos de elementos de formulário usados com frequência (Fonte: <https://www.tutorialrepublic.com/html-tutorial/html-forms.php>)

Mais informações sobre outros atributos podem ser encontradas [aqui](#).

HTML iFrame

Um iframe (ou *frame embutido*) é usado para exibir objetos externos dentro de uma página da web, incluindo outras páginas da web. Um iframe funciona como um mini navegador da web dentro de um navegador da web. Da mesma forma, o conteúdo dentro de um iframe ocorre separado dos elementos adjacentes.

A sintaxe básica para adicionar este recurso a uma página da web é a seguinte:

```
<iframe src="URL"></iframe>
```

A URL especificada no atributo **src** indica a localização de um objeto externo ou de uma página da web.

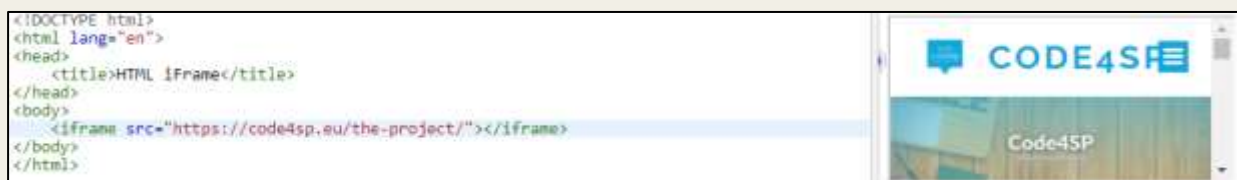


Figura 59 – Código para configurar um frame embutido (Fonte: Autor)

A **altura** e a **largura** do iframe podem ser definidas aplicando o código disposto na Figura 60 abaixo. Os valores dos atributos largura e altura são estipulados em pixels por padrão, mas os utilizadores também podem definir esses valores em percentagem, como 50%, 100%, etc. A largura padrão de um iframe é 300 pixels, enquanto a altura padrão é 150 pixels.

| | |
|--|--|
| <pre><!DOCTYPE html> <html lang="en"> <head> <title>Specify IFrame Dimensions in HTML</title> </head> <body> <h2>Specify Width and Height Using Attributes</h2> <iframe src="/examples/html/hello.html" width="400" height="200"></iframe> <h2>Specify Width and Height Using CSS</h2> <iframe src="/examples/html/hello.html" style="width: 400px; height: 200px;"></iframe> </body> </html></pre> | <h3>Specify Width and Height Using Attributes</h3> <div data-bbox="979 259 1385 465"> <h2>Hello World</h2> <p>This HTML document is embedded inside the current document using an iframe.</p> </div> <hr/> <h3>Specify Width and Height Using CSS</h3> <div data-bbox="979 546 1385 752"> <h2>Hello World</h2> <p>This HTML document is embedded inside the current document using an iframe.</p> </div> |
|--|--|

Figura 60 – Código para configurar a altura e a largura de um quadro embutido (Fonte: Autor)

Como pode ser verificado, o iframe tem uma borda ao contorno dele definida por padrão. Para modificá-lo ou removê-lo, a melhor maneira é usar o recurso de **borda CSS** (a ser ensinado no tópico CSS).

Um iframe também pode ser usado como destino para os hyperlinks. Ele pode ser nomeado usando o atributo **name**. Isso significa que quando uma ligação com um atributo de **destino** (com esse nome definido como valor) é clicada, a fonte vinculada será aberta no mesmo quadro inline, como segue:

| | |
|---|--|
| <pre><!DOCTYPE html> <html lang="en"> <head> <title>Opening links in an IFrame</title> <style> iframe { width: 100%; height: 500px; } </style> </head> <body> <iframe src="https://code4sp.eu/the-project/" name="myframe"></iframe> Open https://code4sp.eu/the-project/en/en/ips </body> </html></pre> |  |
|---|--|

Figura 61 – Usando iframe como destino para um hiperlink (Fonte: Autor)

2.2. Conceitos avançados de HTML

Doctypes HTML

Uma Declaração de Tipo de Documento (DOCTYPE) é uma instrução para o navegador da Web sobre a versão da linguagem de marcação na qual uma página da Web é criada. Ele aparece no topo de uma página da web antes de todos os outros elementos. De acordo com a especificação HTML, todos os documentos HTML requerem uma declaração de tipo de documento válida para garantir que as páginas da Web sejam exibidas da maneira que devem ser. A declaração doctype é frequentemente a primeira coisa definida num documento HTML (mesmo antes da tag de abertura <html>); no entanto, a declaração de doctype em si não é uma tag HTML.

O DOCTYPE para HTML5 é muito curto, conciso e não diferencia maiúsculas de minúsculas: <!DOCTYPE html>.

The following markup can be used as a template to create a new HTML5 document:

```
<!DOCTYPE html> <html lang="en"> <head> <meta charset="utf-8">
<title><!-- Insert your title here --></title> </head> <body> <!--
Insert your content here --> </body> </html>
```

Layouts HTML

Existem diferentes métodos para criar um layout de página web, posicionando os diversos elementos que compõem uma página web de forma bem estruturada e dando uma aparência envolvente ao site. A maioria dos sites geralmente exibe o seu conteúdo em várias linhas e colunas, configuradas como uma revista para fornecer aos utilizadores uma melhor atmosfera de leitura e escrita. Isso pode ser facilmente alcançado utilizando as tags HTML, como <table>, <div>, <header>, <footer>, <section>, etc. e combinando alguns estilos CSS a elas.

A maneira mais simples de criar layouts em HTML é de fato obtida projetando tabelas. Como visto nas seções anteriores, isso geralmente envolve o processo de colocar conteúdo (texto, imagens, etc.) em linhas e colunas.

O layout abaixo contém uma tabela HTML com três linhas e duas colunas. Deve-se observar que a primeira e a última linha abrangem o atributo **colspan**. Deve-se afirmar que o método utilizado para a criação do layout neste exemplo, embora não esteja errado, não é recomendado. Tabelas e estilos embutidos para criar layouts devem ser evitados. Layouts criados usando tabelas são renderizados muito lentamente. **As tabelas só devem ser usadas para exibir dados tabulares**. Para criar tais layouts, **as técnicas de flutuação CSS** são recomendadas. Os usuários devem aprender sobre esta ferramenta além disso



Figura 62 – Layout básico de HTML (Fonte: Autor, adapted from Tutorial Republic. Click [here](#) for better resolution)

O HTML5 estabeleceu novos elementos estruturais, por exemplo **<header>**, **<footer>**, **<nav>**, **<section>**, etc. para identificar as diferentes partes de uma página web de uma forma mais semântica. Este exemplo aplica os novos elementos estruturais HTML5, para criar o mesmo layout criado em Figura 62.

Para saber mais sobre as tags recém-introduzidas, os alunos devem visitar [esta fonte](#).

Cabeçalho HTML

O elemento head é o recetáculo para todos os elementos head que fornecem informações extras sobre o documento ou referência a outros recursos necessários para que o documento funcione corretamente. Ele ilustra as propriedades do documento, como título, entrega meta-informações como conjunto de caracteres, informa ao navegador onde encontrar as folhas de estilo ou scripts que permitem expandir o documento HTML de forma interativa.

Os elementos HTML que podem ser usados dentro do elemento `<head>` são: `<title>`, `<base>`, `<link>`, `<style>`, `<meta>`, `<script>` e `<noscript>`.

The HTML title Element

O elemento `<title>` identifica o título do documento e apenas um é necessário em todos os documentos HTML/XHTML para produzir um documento válido. Deve ser colocado dentro do elemento `<head>`. O elemento title compreende texto simples e entidades e não pode incluir outras tags de marcação. Pode ser usado para diferentes funções:

- Para mostrar um título na barra de título do navegador e na barra de tarefas;
- Para entregar um título para a página quando ela é adicionada aos favoritos ou marcada;
- Para apresentar um título para a página nos resultados do mecanismo de pesquisa (por exemplo, pesquisa do Google).

Deve ser curto e específico para o conteúdo do documento, pois os motores de busca devem prestar atenção especial às palavras presentes no título – idealmente, deve ter 65 caracteres.

Um título em um documento HTML deve ser exibido da seguinte forma:

```
<!DOCTYPE html> <html lang="en"> <head> <title>A simple HTML document</title> </head> <body> <p>Hello World!</p> </body> </html>
```

The HTML base Element

The HTML `<base>` element is used to identify a base URL for all relative links contained in the document. For example, one can set the base URL once at the top of the page, and then all subsequent relative links will use that URL as a starting point, as follows:

```
<!DOCTYPE html> <html lang="en"> <head> <title>Defining a base URL</title> <base href=" https://code4sp.eu/the-project/" > </head> <body> <p><a href=" https://code4sp.eu/the-project/" >HTML Head</a>.</p> </body> </html>
```



Figura 63 – Elemento básico HTML (Fonte: Autor)

O elemento HTML `<base>` deve ser localizado antes de qualquer elemento que pertença a um recurso externo. HTML permite apenas um elemento base para cada documento.

Elemento de ligação HTML

O elemento `<link>` descreve a correlação entre o documento atual e um documento ou recurso externo. Um uso comum do elemento link é conectar-se a folhas de estilo externas.

| | |
|---|---|
| <pre><!DOCTYPE html> <html lang="en"> <head> <title>Linking Style Sheets</title> <link rel="stylesheet" href="/examples/css/style.css"> </head> <body> <h1>Linking style sheets</h1> <p>The styles of this HTML document are defined in linked style sheet.</p> </body> </html></pre> | <h2 style="color: orange;">Linking style sheets</h2> <p>The styles of this HTML document are defined in linked style sheet.</p> |
|---|---|

Figura 63 – HTML base element (Fonte: <https://www.tutorialrepublic.com/codelab.php?topic=html&file=linking-style-sheet>)

Deve ser declarado que o elemento <head> de um documento HTML pode incluir qualquer número de elementos <link>. O elemento <link> tem atributos, mas nenhum conteúdo.

O Elemento Estilo de HTML

O elemento <style> é aplicado para descrever informações de estilo incorporadas para um documento HTML. As regras de estilo dentro do elemento <style> indicam como os elementos HTML devem ser renderizados num navegador. Uma folha de estilo incorporada deve ser usada quando um único documento tem um estilo único. Se a mesma folha de estilo for usada em vários documentos, uma folha de estilo externa seria mais adequada.


| | |
|--|---|
| <pre><!DOCTYPE html> <html lang="en"> <head> <title>Code4SP</title> <style> body { background-color: Blue; } h1 { color: white; } p { color: white; } </style> </head> <body> <h1>CODE4SP</h1> <p>Coding for social promotion.</p> </body> </html></pre> |  |
|--|---|

Figura 64 – Codificar vários elementos de estilo (Fonte: Autor)

O Elemento Meta de HTML

O elemento `<meta>` fornece metadados sobre o documento HTML, que é um conjunto de dados que descreve e fornece informações sobre outros dados. As tags `<meta>` aparecem sempre dentro do elemento `<head>` e normalmente são usadas para especificar o conjunto de caracteres, descrição da página, palavras-chave, autor do documento e configurações da janela de visualização.


| | |
|--|---|
| <pre><!DOCTYPE html> <html lang="en"> <head> <title>Code4SP</title> <meta charset="utf-8"> <meta name="author" content="CODE4SP project team"> </head> <body> <h1>Code4SP</h1> <p>Coding for social promotion.</p> </body> </html></pre> |  |
|--|---|

Figura 64 – O elemento meta (Fonte: Autor)

Como visto acima, as metatags incluem informações sobre uma página da web. Não é visível no navegador (embora seja analisável por máquina). Os metadados são utilizados por navegadores (como exibir conteúdo ou recarregar a página), mecanismos de pesquisa (palavras-chave) e outros serviços da web. Além disso, existe uma técnica para permitir que os web designers governem a **viewport**, por meio da tag `<meta>`. A janela de visualização é a área visível do usuário de uma página da web. Ele difere de dispositivo para dispositivo (será maior num ecrã de computador do que num telemóvel).

O elemento `<meta>` a seguir deve ser incluído em todas as páginas da web, pois dará ao navegador diretrizes sobre como assumir as dimensões e o dimensionamento da página:

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

A parte `width=device-width` define a largura da página para seguir a largura da tela do dispositivo (que varia de acordo com a tela). A parte `initial-scale=1.0` define o nível de zoom inicial quando a página é carregada inicialmente pelo navegador.

A diferença entre páginas da web com ou sem meta tag viewport é bem visível no exemplo a seguir:

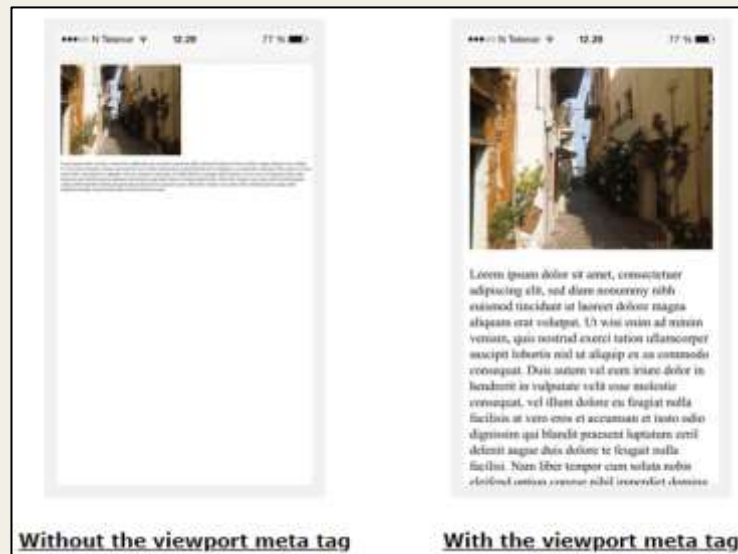


Figura 65 – Exemplos de metatags com ou sem viewport (Fonte: https://www.w3schools.com/tags/tag_meta.asp)

O elemento script HTML

O elemento `<script>` é usado para definir o script do lado do cliente, como JavaScript em documentos HTML.



Figura 65 – O elemento script (Fonte: Autor)

Trabalhar com script do lado do cliente

O script do lado do cliente está vinculado ao tipo de programas de computador executados pelo navegador da Web do utilizador. **JavaScript (JS)** é a linguagem de script do lado do cliente mais difundida na web. O elemento `<script>` é usado para incorporar ou referenciar JavaScript num documento HTML para adicionar interatividade a páginas da Web e criar uma experiência mais amigável. Alguns dos usos mais comuns do JavaScript são validação de formulários, geração de mensagens de alerta, criação de galeria de imagens, exibição de conteúdo oculto, manipulação de DOM, etc.

JavaScript pode ser incorporado de duas formas:

1. Diretamente dentro da página HTML;
2. Colocado num arquivo de script externo e **referenciado** dentro da página HTML.

Nota: ambas as técnicas usam o elemento `<script>`.

Para embutir JS num arquivo HTML, o usuário deve adicionar o código como conteúdo do elemento `<script>`, seguindo o exemplo da Figura 66. De preferência, os elementos script devem ser colocados no final da página, antes da tag de fecho do corpo (`</body>`), pois quando o navegador encontra um script, ele pausa a renderização do restante da página até desconstruir o script que pode impactar drasticamente o desempenho do site.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Embedding JavaScript</title>
</head>
<body>
  <div id="greet"></div>
  <script>
    document.getElementById("greet").innerHTML = "Code4SP";
  </script>
</body>
</html>
```

Figura 66 – Incorporando JS em um documento HTML (Fonte: Autor)

Os códigos JavaScript também podem ser colocados num arquivo separado, chamando uma extensão **.js**, enquanto a corresponde no documento HTML usando o atributo **src** da tag **<script>**. Isso pode ser especialmente útil se o web designer quiser disponibilizar o mesmo script para vários documentos, para que ele não precise executar as mesmas tarefas repetidamente. Quando o atributo **src** é indicado, o elemento **<script>** deve estar vazio, para que o web designer não possa usar o mesmo elemento **<script>** para incorporar o JavaScript e vincular a um arquivo JavaScript externo num documento HTML.

Se um navegador, por algum motivo, não suportar scripts do lado do cliente ou os utilizadores tiverem desabilitado o JS em seus navegadores, o elemento **<no script>** poderá ser usado para fornecer um conteúdo alternativo. Este elemento pode incluir qualquer elemento HTML, exceto **<script>**, pois pode ser incluído no elemento **<body>** de uma página HTML.

Entidades HTML

A maioria dos alunos certamente ficará curiosa sobre como exibir caracteres e símbolos especiais em seus processos de programação. Assim, este subcapítulo pretende explicar como eles podem ser bem sucedidos em fazer tal coisa.

Primeiro, é importante entender o que é uma entidade HTML. Conforme percebido nos capítulos anteriores, alguns caracteres são bastante reservados em HTML. Por exemplo, não se pode usar sinais como '<' ou '>', pois o navegador pode confundi-los com uma marcação. Além disso, alguns caracteres simplesmente não estão disponíveis no teclado (por exemplo, o símbolo de copyright).

Esses caracteres especiais podem ser exibidos simplesmente sendo substituídos pelas entidades de caracteres (ou apenas entidades), resolvendo os problemas mencionados acima. Abaixo está uma lista das entidades mais usadas em HTML:

| Resultado | Descrição | Nome da entidade | Referência numérica |
|-----------|-------------------|------------------|---------------------|
| | Espaço sem quebra | | |
| < | Menos do que | < | < |
| > | Mais do que | > | > |
| & | ampersand | & | & |
| " | marca citação | " | " |
| ' | apóstrofe | ' | ' |
| ¢ | cêntimo | ¢ | ¢ |

| | | | |
|---|-----------------|---------|---------|
| £ | pound | £ | £ |
| ¥ | yen | ¥ | ¥ |
| € | euro | € | € |
| © | copyright | © | © |
| ® | marca registada | ® | ® |
| ™ | trademark | ™ | ™ |

Table 3 – As entidades mais frequentes em HTML (Fonte: <https://www.tutorialrepublic.com/html-tutorial/html-entities.php>)

Referências de caracteres numéricos também podem ser usadas como alternativa para nomes de entidades, especialmente porque têm suporte mais forte do navegador e podem ser usadas para especificar qualquer caracter Unicode, porém as entidades são limitadas a um subgrupo deste.

URL HTML

URL: quantas vezes essa abreviação apareceu em ambientes virtuais? Significa Uniform Resource Locator e funciona como o endereço global de documentos e outros recursos na World Wide Web. O seu objetivo é identificar a localização de um documento e outros recursos disponíveis online, especificando o mecanismo para lhe aceder usando um navegador da web. Por exemplo, <https://code4sp.eu/> é um URL.

A sintaxe geral de URLs pode ser descrita da seguinte forma:
`scheme://host:port/path?query-string#fragment-id`

Os URLs têm uma estrutura linear e são compostos pelos seguintes componentes:

- **Nome do esquema** — O esquema reconhece o protocolo a ser usado para acessar a um recurso na Internet. Os nomes dos esquemas são seguidos pelos três caracteres `://` (dois pontos e duas barras). Os protocolos mais usados são `http://`, `https://`, `ftp://` e `mailto://`.
- **Nome de anfitrião** — identifica o host onde o recurso está situado. Um nome de host é um nome de domínio dado a um computador host. Isso geralmente é uma combinação do nome local do host com o nome do parente domain. Por exemplo, `www.code4sp.eu/` consiste no nome da máquina do host `www` e no nome de domínio `code4sp.eu`.
- **Número do post** — Os servidores geralmente entregam mais do que um tipo de serviço, portanto, eles devem ser informados sobre qual serviço está sendo solicitado. Essas solicitações são feitas pelo número da porta. Normalmente, os números de porta conhecidos de um serviço são omitidos da URL. Por exemplo, HTTP de serviço da web é executado por padrão na porta 80, HTTPS é executado por padrão na porta 443.
- **Caminho** — identifica o recurso específico dentro do host que o utilizador deseja acessar. Por exemplo, `/html/html-url.php`, `/news/technology/`, etc.
- **String de consulta** — contém dados a serem passados para scripts do lado do servidor, executados no servidor web. Por exemplo, parâmetros para uma pesquisa. A string de consulta precedida por um ponto de interrogação (?), geralmente é uma string de pares de nome e valor separados por e comercial (&), por exemplo, `?first_name=John&last_name=Corner, q=mobile+phone`, e por aí fora.
- **Identificador de fragmentos** — especifica um local dentro da página. O navegador move-se para exibir essa parte da página. O identificador de fragmento introduzido por um caracter hash (#) é a última parte opcional de um URL para um documento.

Codificação do URL HTML

Sabe-se que às vezes os dados não são transmitidos com segurança pela Internet. Isso acontece principalmente porque os URLs não são codificados de forma completa ou precisa e causa alguns mal-entendidos entre os utilizadores da Internet.

De acordo com a RFC 3986, os caracteres numa URL restringem-se apenas a um conjunto definido de caracteres US-ASCII reservados e não reservados. Quaisquer outros caracteres não são permitidos num URL (é por isso que alguns caracteres latinos e cirílicos não são vistos no URL). Mas o URL geralmente inclui caracteres fora do conjunto de caracteres US-ASCII, portanto, eles devem ser convertidos num formato US-ASCII válido para interoperabilidade mundial. A codificação de URL é um processo de conversão de informações de URL para que possam ser transmitidas com segurança pela Internet.

Para mapear a grande variedade de caracteres usados globalmente, um método de duas etapas é seguido:

- Em primeiro lugar, os dados são codificados em relação à codificação de caracteres UTF-8.
- Então, apenas os bytes que não correspondem aos caracteres no conjunto não reservado devem ser codificados por percentagem como %HH, onde HH é o valor hexadecimal do byte.

Por exemplo, veja esta popular frase portuguesa:

“Quem vê caras, não vê corações.” [“Faces we see, hearts we do not know.”]

Esta frase seria codificada como:

Quem%20v%C3%AA%20caras%2C%20n%C3%A3o%20v%C3%AA%20cora%C3%A7%C3%B5es.

Ç, ç (c-cedilla) is a Latin script letter, as well as well as the accents used (~ and ^).

Certos caracteres são restritos ao uso numa URL, pois podem (ou não) ser definidos como delimitadores pela sintaxe genérica num determinado esquema URL (por exemplo, barra / caracteres são usados para separar diferentes partes de um URL).

Caracteres reservados num URL são os seguintes:

| | | | | | | | | | | | | | | | | | |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| ! | # | \$ | & | ' | (|) | * | + | , | / | : | ; | = | ? | @ | [|] |
| %21 | %23 | %24 | %26 | %27 | %28 | %29 | %2A | %2B | %2C | %2F | %3A | %3B | %3D | %3F | %40 | %5B | %5D |

Figura 67 – Caracteres reservados num URL (Fonte: <https://www.tutorialrepublic.com/html-tutorial/html-url-encode.php>)

No entanto, também existem caracteres que, apesar de permitidos num URL, não possuem uma finalidade reservada – por isso são chamados de “caracteres não reservados”.

Isso inclui letras maiúsculas e minúsculas, dígitos decimais, hífen, ponto, sublinhado e til. A tabela a seguir lista todos os caracteres não reservados num URL:

| | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |
| a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p | q | r | s | t | u | v | w | x | y | z |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | - | _ | . | ~ | | | | | | | | | | | | |

Figura 68 – Caracteres não reservados em um URL (Fonte: <https://www.tutorialrepublic.com/html-tutorial/html-url-encode.php>)

Para codificar caracteres, os utilizadores podem usar [este conversor](#).

Validação HTML

Para garantir que um código HTML segue os padrões atuais da Web, livre de erros, é importante descobrir como validar um código HTML. Os iniciantes geralmente cometem

erros ao escrever códigos HTML, e códigos incorretos ou fora do padrão certamente causarão resultados inesperados na forma como uma página da Web é exibida num navegador.

Para evitar que isso aconteça, os utilizadores podem testar seus códigos dentro das diretrizes e padrões formais estabelecidos pela Wide Web Consortium (W3C) para páginas da web HTML/XHTML. Existe uma [ferramenta online](#) que verifica automaticamente os códigos HTML e aponta quaisquer problemas/erros, como tags de fecho ausentes ou aspas ausentes em torno de atributos.

O processo de validação de uma página web garante o respeito às normas/padrões definidos pelo W3C, por isso é muito importante. Algumas razões para validar uma página da web são:

- Ajuda a criar páginas da Web compatíveis com vários navegadores e plataformas. Muito provavelmente eles serão compatíveis com a versão futura dos navegadores e padrões da web.
- As páginas da web em conformidade com os padrões aumentam a visibilidade dos spiders e rastreadores do mecanismo de pesquisa, como resultado, as páginas da web provavelmente aparecerão nos resultados da pesquisa.
- Isso reduzirá erros inesperados e tornará suas páginas da web mais acessíveis ao visitante.

Características HTML5

Neste subcapítulo, serão explicados os recursos da quinta (e última) versão HTML principal recomendada pelo W3C.

Novos tipos de entrada HTML5

O HTML5 apresenta vários novos tipos de `<input>` como email, data, hora, cor, intervalo e assim por diante, com o objetivo de melhorar a experiência do usuário e tornar os formulários mais interativos. No entanto, se um navegador não reconhecer esses novos tipos de entrada, ele vai considerá-los uma caixa de texto normal.

Os seguintes são alguns novos tipos de entrada:

- cor
- data
- datetime-local
- email
- mês
- número
- range
- pesquisa
- tel
- hora
- url
- semana

Com relação à entrada de `cores`, permite selecionar uma cor de um seletor de cores e fornece informações sobre o valor da cor em formato hexadecimal (por exemplo, #000000, que é preto, a cor padrão se o usuário não especificar um valor), conforme verificado na *Figura 69* abaixo.

Deve-se notar que a entrada de cores é suportada em todos os principais navegadores modernos (Firefox, Chrome, Opera, Safari (12.1+), Edge (14+)), mas não é suportado pelo Microsoft Internet Explorer e versões mais antigas dos navegadores Apple Safari.

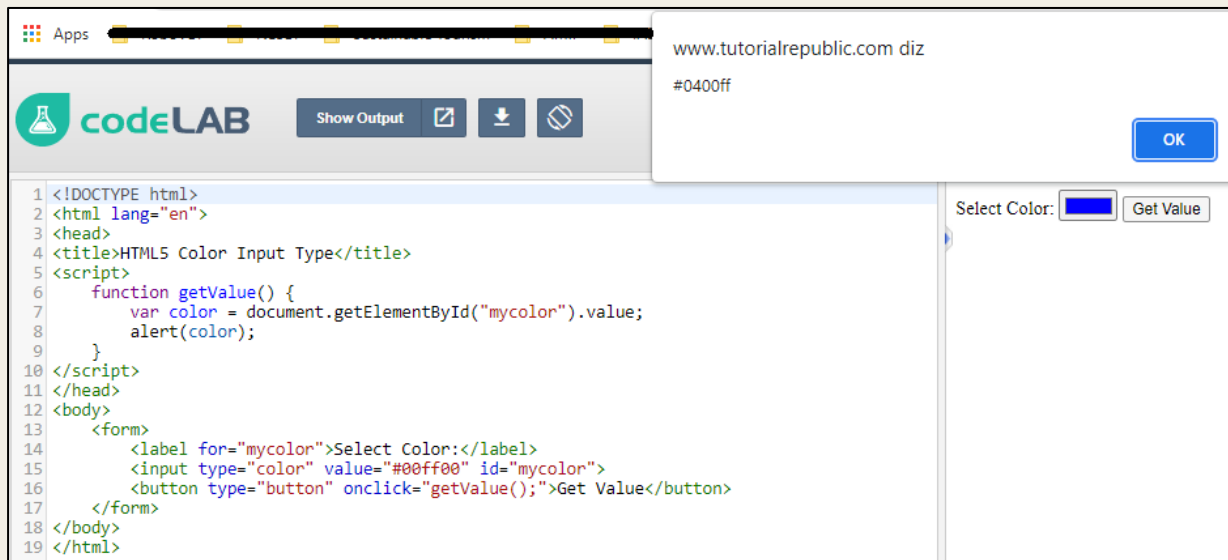


Figura 69 – Escolher uma cor usando o color input (Fonte: <https://www.tutorialrepublic.com/codelab.php?topic=html5&file=color-input-type>)

Quanto ao tipo de entrada de data, permite ao usuário selecionar uma data de um calendário suspenso, no qual pode escolher o ano, mês e dia (mas não a hora). Esse recurso também é compatível com a maioria dos navegadores, exceto Internet Explorer e Safari.

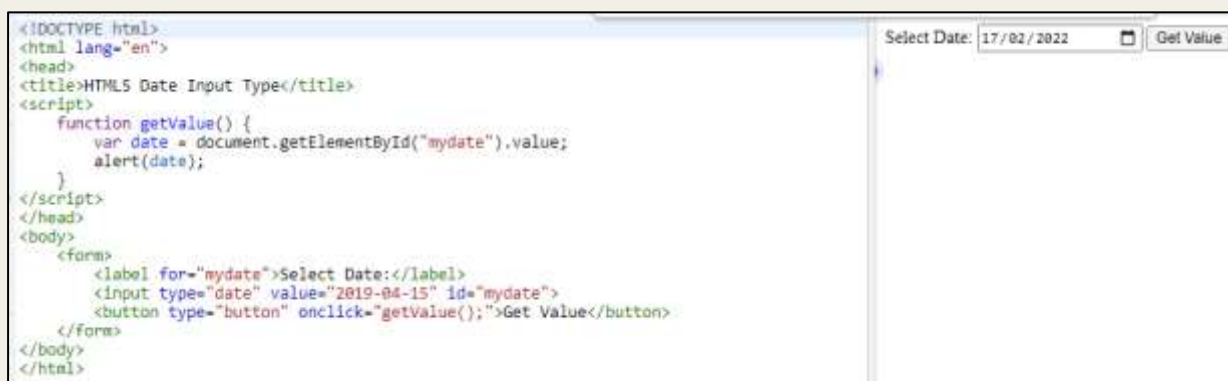
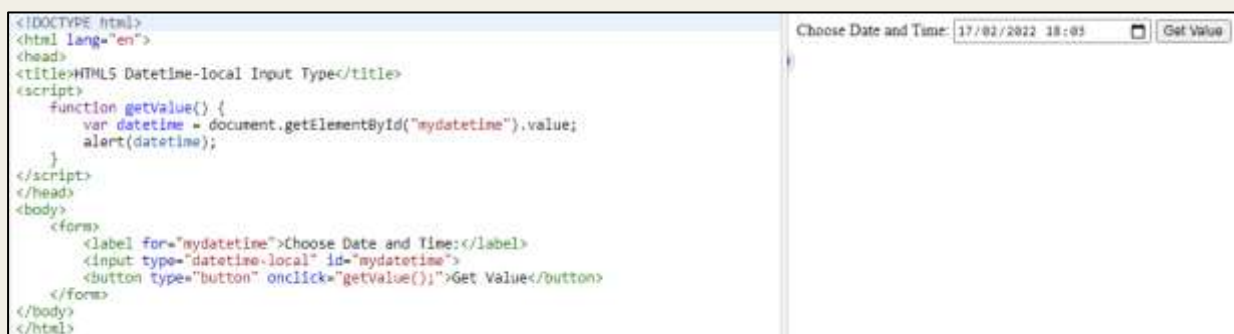


Figura 70 – Data input type (Fonte: <https://www.tutorialrepublic.com/codelab.php?topic=html5&file=color-input-type>)

O tipo de entrada **datetime-local** disponibiliza ao utilizador a seleção de data e hora local, incluindo ano, mês e dia, e incluindo a hora em horas e minutos. Esta entrada é suportada pelo Safari, Firefox e IE, mas não pelo Chrome, Edge e Opera.



```

<!DOCTYPE html>
<html lang="en">
<head>
<title>HTML5 Datetime-Local Input Type</title>
<script>
function getValue() {
var datetime = document.getElementById("mydatetime").value;
alert(datetime);
}
</script>
</head>
<body>
<form>
<label for="mydatetime">Choose Date and Time:</label>
<input type="datetime-local" id="mydatetime">
<button type="button" onclick="getValue()">Get Value</button>
</form>
</body>
</html>

```

Figura 71 – A entrada datetime-local (Fonte: <https://www.tutorialrepublic.com/codelab.php?topic=html5&file=color-input-type>)

Para permitir que o utilizador insira o seu endereço de e-mail, **e-mail** é o tipo de entrada preferencial. É como um tipo de entrada de texto padrão, mas se for aplicado em combinação com o atributo obrigatório, os navegadores podem procurar os padrões para garantir que um endereço de e-mail formatado corretamente seja inserido. O campo de entrada de email pode ser estilizado para diferentes estados de validação, quando um valor é inserido usando as pseudoclasses **:valid**, **:invalid** ou **:required**. Este tipo de entrada é suportado por todos os principais navegadores.



Figura 72 – A entrada de email (Fonte: <https://www.tutorialrepublic.com/html-tutorial/html5-new-input-types.php>)

No que diz respeito ao tipo de entrada de **mês**, esta é uma funcionalidade muito semelhante às anteriores, pois permite seleccionar um mês e ano de um calendário suspenso (sendo 'YYYY' o ano e "MM" o mês). Deve-se notar que isso não é suportado pelo Firefox, Safari e Internet Explorer. Apenas os navegadores Chrome, Edge e Opera o suportam.

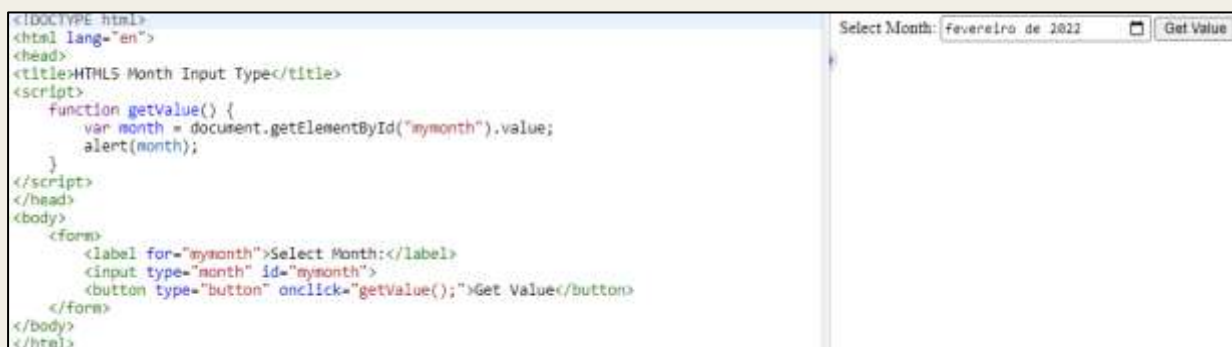


Figura 73 – A entrada de mês (Fonte: <https://www.tutorialrepublic.com/html-tutorial/html5-new-input-types.php>)

No que diz respeito ao tipo de entrada **numérica**, é utilizado para inserir um valor numérico. O web designer também pode limitar o utilizador a inserir apenas valores aceitáveis. Para que isso aconteça, os atributos adicionais **min**, **max** e **step** devem ser usados. Este recurso é suportado por todos os principais navegadores da web.

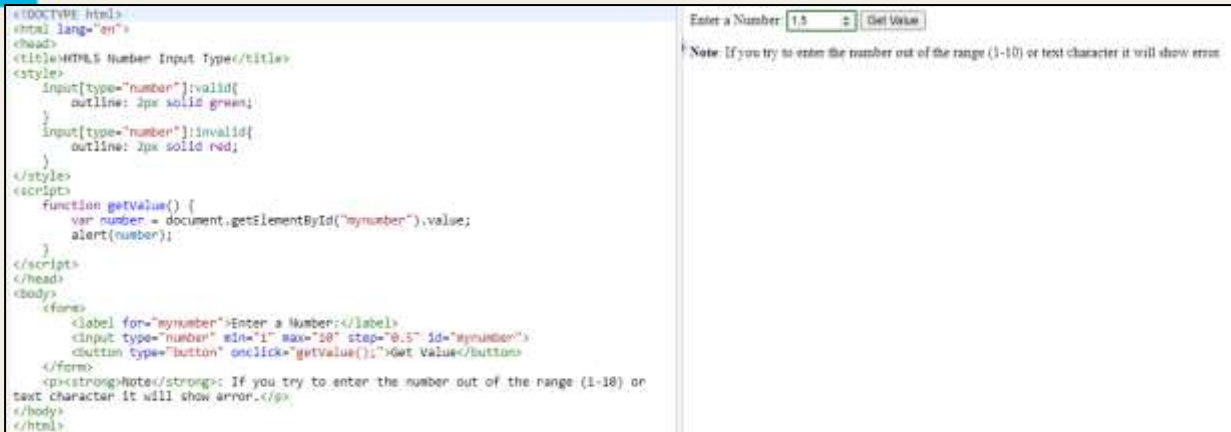


Figura 74 – A entrada numérica (Fonte: <https://www.tutorialrepublic.com/html-tutorial/html5-new-input-types.php>)

O tipo de entrada de **intervalo** pode ser aplicado para registrar um valor numérico dentro de um intervalo específico. Funciona de maneira muito semelhante à entrada numérica acima, embora apresente uma maneira mais fácil de inserir um **número**. Este tipo de entrada é suportado por todos os principais navegadores da web.

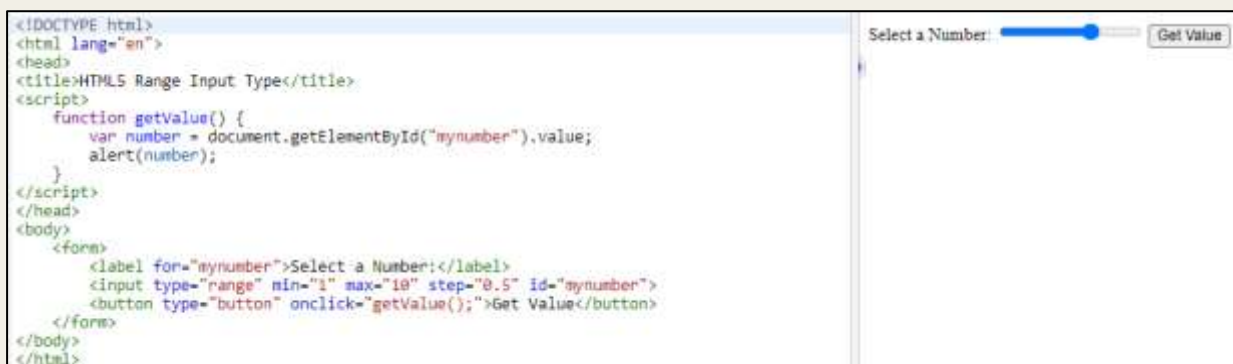


Figura 75 – A entrada de range (Fonte: <https://www.tutorialrepublic.com/html-tutorial/html5-new-input-types.php>)

O tipo de entrada de **pesquisa** é adequado para gerar campos de entrada de pesquisa. Deve-se dizer que, em alguns navegadores (isto é, Chrome e Safari), assim que o utilizador começa a escrever na caixa de pesquisa, surge uma minúscula cruz do lado direito do campo, que pode ser útil para limpar todo o campo de pesquisa. É suportado por todos os principais navegadores.

```
<!DOCTYPE html>
<html lang="en">
<head>
<title>HTML5 Search Input Type</title>
<script>
function getValue() {
var term = document.getElementById("mysearch").value;
alert(term);
}
</script>
</head>
<body>
<form>
<label for="mysearch">Search Website:</label>
<input type="search" id="mysearch" placeholder="Type something...">
<button type="button" onclick="getValue();">Get Value</button>
</form>
</body>
</html>
```

Figura 76 – A entrada de pesquisa (Fonte: <https://www.tutorialrepublic.com/html-tutorial/html5-new-input-types.php>)

Quanto ao tipo de entrada **tel**, é especialmente útil para inserir um número de telefone. Como os navegadores não suportam a validação de entrada **tel** por padrão, o atributo **placeholder** pode ser usado para ajudar os utilizadores a inserir o formato correto para seu número de telefone ou indicar uma expressão regular para validar a entrada do usuário aplicando o atributo **pattern**. Este recurso não é suportado por nenhum navegador, pois os números de telefone variam muito em todo o mundo.

```
<!DOCTYPE html>
<html lang="en">
<head>
<title>HTML5 Tel Input Type</title>
<script>
function getValue() {
var phone = document.getElementById("myphone").value;
alert(phone);
}
</script>
</head>
<body>
<form>
<label for="myphone">Telephone Number:</label>
<input type="tel" id="myphone" placeholder="xx-xxxx-xxxx" required>
<button type="button" onclick="getValue();">Get Value</button>
</form>
</body>
</html>
```

Figura 77 – A entrada tel (Fonte: <https://www.tutorialrepublic.com/html-tutorial/html5-new-input-types.php>)

Em relação ao tipo de entrada de **hora**, ele pode ser usado para inserir qualquer hora (horas e minutos), e o navegador pode usar os dois formatos de hora (12 ou 24 horas) para inserir horas, dependendo da região. Esta entrada não é suportada pelos navegadores IE e Safari.

```

<!DOCTYPE html>
<html lang="en">
<head>
<title>HTML5 Time Input Type</title>
<script>
function getValue() {
var time = document.getElementById("mytime").value;
alert(time);
}
</script>
</head>
<body>
<form>
<label for="mytime">Select Time:</label>
<input type="time" id="mytime">
<button type="button" onclick="getValue();">Get Value</button>
</form>
</body>
</html>

```

Figura 78 – A entrada de hora (Fonte: <https://www.tutorialrepublic.com/html-tutorial/html5-new-input-types.php>)

Em relação ao tipo de entrada de **url**, ele pode ser usado para inserir URLs ou endereços da web. O atributo `multiple` pode ser usado para inserir mais do que uma URL. Além disso, se o atributo obrigatório for restrito, o navegador executará automaticamente a validação para garantir que apenas o texto que atenda ao formato padrão para URLs entre na caixa de entrada. Todos os principais navegadores suportam este tipo de entrada.

```

<!DOCTYPE html>
<html lang="en">
<head>
<title>HTML5 URL Input Type</title>
<style>
input[type="url"]:valid{
outline: 2px solid green;
}
input[type="url"]:invalid{
outline: 2px solid red;
}
</style>
<script>
function getValue() {
var url = document.getElementById("myurl").value;
alert(url);
}
</script>
</head>
<body>
<form>
<label for="myurl">Enter Website URL:</label>
<input type="url" id="myurl" required>
<button type="button" onclick="getValue();">Get Value</button>
</form>
<p><strong>Note</strong>: Enter URL in the form like https://www.google.com</p>
</body>
</html>

```

Figura 79 – A entrada de URL (Fonte: <https://www.tutorialrepublic.com/html-tutorial/html5-new-input-types.php>)

Por fim, o tipo de entrada de **semana** permite que o usuário escolha uma semana e um ano em um calendário suspenso. Este recurso não é suportado pelo Firefox, Safari e IE, mas atualmente é suportado pelos navegadores Chrome, Edge e Opera.

```

<!DOCTYPE html>
<html lang="en">
<head>
<title>HTML5 week Input Type</title>
<script>
function getValue() {
var week = document.getElementById("myweek").value;
alert(week);
}
</script>
</head>
<body>
<form>
<label for="myweek">Select Week:</label>
<input type="week" id="myweek">
<button type="button" onclick="getValue();">Get Value</button>
</form>
</body>
</html>

```

Figura 80 – A entrada de semana (Fonte: <https://www.tutorialrepublic.com/html-tutorial/html5-new-input-types.php>)

Canvas HTML5

Este subcapítulo será essencialmente útil para aprender a desenhar gráficos usando o elemento canvas do HTML5. Ele foi originalmente criado pela Apple para os widgets do painel do Mac OS e para habilitar gráficos no Safari. Além disso, foi adotado por outros navegadores, como Firefox, Google Chrome e Opera.

Por padrão, o elemento `<canvas>` tem 300px de largura e 150px de altura sem nenhuma borda e conteúdo. No entanto, largura e altura personalizadas podem ser especificadas usando o recurso CSS altura e largura.

O ecrã é uma área bidimensional de quatro lados. As coordenadas do canto superior esquerdo do ecrã são (0, 0), identificadas como origem, e as coordenadas do canto inferior direito são (largura do ecrã, altura do ecrã), como pode ser visto usando a ferramenta interativa disponível [aqui](#).

Para desenhar caminhos e formas básicas utilizando o elemento canvas HTML5 recentemente introduzido e JavaScript, será útil dar uma olhada a vários modelos.

Em primeiro lugar, o modelo base para desenhar caminhos e formas na tela 2D HTML5:



```

1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4 <meta charset="utf-8">
5 <title>Embedding Canvas Into HTML Pages</title>
6 <style>
7   canvas {
8     border: 1px solid #000;
9   }
10 </style>
11 <script>
12   window.onload = function() {
13     var canvas = document.getElementById("myCanvas");
14     var context = canvas.getContext("2d");
15     // draw stuff here
16   };
17 </script>
18 </head>
19 <body>
20   <canvas id="myCanvas" width="300" height="200"></canvas>
21 </body>
22 </html>

```

Figura 81 – O modelo básico para desenhar caminhos e formas no ecrã 2D HTML5 (Fonte: <https://www.tutorialrepublic.com/html-tutorial/html5-new-input-types.php>)

Todas as linhas, exceto as de 7 a 11, são bastante diretas e fáceis de interpretar. A função anónima afixada ao evento `window.onload` será executada quando a página for carregada. Assim que a página é carregada, pode-se aceder ao elemento canvas com o método `document.getElementById()`. Mais tarde, um contexto de ecrã 2D é definido passando 2d para o método `getContext()` do objeto do ecrã.

O passo inicial para desenhar no ecrã é uma linha reta. Os procedimentos mais importantes usados para este propósito são `moveTo()`, `lineTo()` e o `stroke()`. O método `moveTo()` identifica a posição do cursor de desenho na tela, enquanto o método `lineTo()` é usado para definir as coordenadas do ponto final da linha e, finalmente, o método `stroke()` é usado para tornar a linha visível:



```

1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4 <meta charset="utf-8">
5 <title>Drawing a Line on the Canvas</title>
6 <style>
7   canvas {
8     border: 1px solid #000;
9   }
10 </style>
11 <script>
12   window.onload = function() {
13     var canvas = document.getElementById("myCanvas");
14     var context = canvas.getContext("2d");
15     context.moveTo(50, 150);
16     context.lineTo(250, 50);
17     context.stroke();
18   };
19 </script>
20 </head>
21 <body>
22   <canvas id="myCanvas" width="300" height="200"></canvas>
23 </body>
24 </html>

```

Figura 82 – Os procedimentos `moveTo()`, `lineTo()` e `stroke()` (Fonte: <https://www.tutorialrepublic.com/html-tutorial/html5-new-input-types.php>)

Que tal **desenhar um arco**? Ele pode ser obtido simplesmente usando o método `arc()`, cuja sintaxe é:

```
context.arc(centerX, centerY, radius, startingAngle, endingAngle, counterclockwise);
```

No exemplo acima, um arco foi desenhado na tela inserindo um código JavaScript:

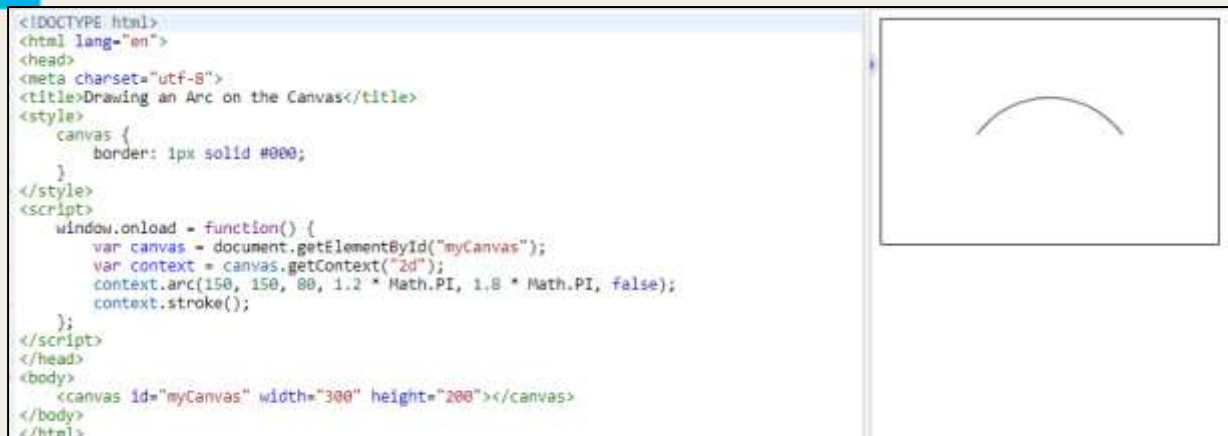


Figura 83 – Desenhar um arco usando o código JS (Fonte: <https://www.tutorialrepublic.com/html-tutorial/html5-new-input-types.php>)

Para desenhar um retângulo e formas quadradas, o método `rect()` é o caminho a seguir. Ela envolve quatro parâmetros: posições x, y do retângulo e sua largura e altura. A sintaxe básica do método `rect()` é a seguinte:

```
context.rect(x, y, width, height);
```

Para desenhar usando um código JS:



Figura 84 – Desenhar um retângulo usando um código JS (Fonte: <https://www.tutorialrepublic.com/html-tutorial/html5-new-input-types.php>)

Ao contrário do método `rect()`, não existe um procedimento específico para desenhar um círculo. No entanto, o resultado pode ser obtido criando um arco totalmente fechado, usando o método `arc()`. A sintaxe para desenhar um círculo completo usando o método `arc()` é a seguinte:

```
context.arc(centerX, centerY, radius, 0, 2 * Math.PI, false);
```



Figura 85 – Desenhando um círculo na canvas HTML5 (Fonte: <https://www.tutorialrepublic.com/html-tutorial/html5-new-input-types.php>)

Em relação aos **estilos e cores do traço**, a cor padrão do traço é preto, sendo sua espessura de 1 pixel. No entanto, esses atributos podem ser alterados usando as propriedades `strokeStyle` e `lineWidth`, conforme segue na *Figura 86*.

Na *Figura 87*, é possível definir o estilo de cap para as linhas usando a propriedade `lineCap`, com três estilos disponíveis: `butt`, `round` e `square`.

Também é possível preencher a cor dentro das formas da canvas usando a abordagem `fillStyle()`. A *Figura 88* mostra como preencher uma cor sólida dentro de uma forma retangular. Ao projetar as formas na tela, sugere-se usar o método `fill()` antes do método `stroke()` para renderizar o traçado adequadamente.

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Setting Stroke Color and Width on the Canvas</title>
<style>
  canvas {
    border: 1px solid #000;
  }
</style>
<script>
  window.onload = function() {
    var canvas = document.getElementById("myCanvas");
    var context = canvas.getContext("2d");
    context.lineWidth = 5;
    context.strokeStyle = "orange";
    context.moveTo(50, 150);
    context.lineTo(250, 50);
    context.stroke();
  };
</script>
</head>
<body>
  <canvas id="myCanvas" width="300" height="200"></canvas>
</body>
</html>
```

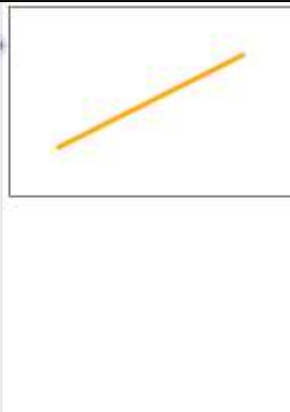


Figura 86 – Definir os estilos e cores no traçado usando as propriedades strokeStyle e lineWidth (Source: <https://www.tutorialrepublic.com/html-tutorial/html5-new-input-types.php>)

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Setting Stroke Cap Style on the Canvas</title>
<style>
  canvas {
    border: 1px solid #000;
  }
</style>
<script>
  window.onload = function() {
    var canvas = document.getElementById("myCanvas");
    var context = canvas.getContext("2d");
    context.lineWidth = 10;
    context.strokeStyle = "orange";
    context.lineCap = "round";
    context.arc(150, 150, 80, 1.2 * Math.PI, 1.8 * Math.PI, false);
    context.stroke();
  };
</script>
</head>
<body>
  <canvas id="myCanvas" width="300" height="200"></canvas>
</body>
</html>
```



Figura 87 – Configurar o estilo de cap para as linhas usando a propriedade lineCap (Source: <https://www.tutorialrepublic.com/html-tutorial/html5-new-input-types.php>)

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Filling Color Inside a Rectangle on the Canvas</title>
<style>
  canvas {
    border: 1px solid #000;
  }
</style>
<script>
  window.onload = function() {
    var canvas = document.getElementById("myCanvas");
    var context = canvas.getContext("2d");
    context.rect(50, 50, 200, 100);
    context.fillStyle = "#808080";
    context.fill();
    context.lineWidth = 5;
    context.strokeStyle = "black";
    context.stroke();
  };
</script>
</head>
<body>
  <canvas id="myCanvas" width="300" height="200"></canvas>
</body>
</html>
```

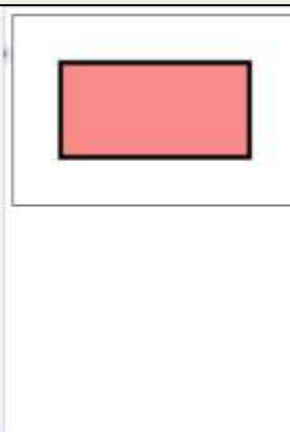


Figura 88 – Arquivar a cor dentro das formas da tela usando a abordagem fillStyle() (Source: <https://www.tutorialrepublic.com/html-tutorial/html5-new-input-types.php>)

Também é possível preencher gradiente de cores (uma transição visual suave de uma cor para outra) nas formas da tela. Existem dois tipos de gradientes aqui: linear e radial.

A sintaxe básica para criar um **gradiente linear** é:

```
var grd = context.createLinearGradient(startX, startY, endX, endY);
```

A sintaxe básica para criar um **gradiente radial** é:

```
var grd = context.createRadialGradient(startX, startY, startRadius, endX, endY, endRadius);
```

A *Figura 89* mostra como preencher uma cor **gradiente linear** dentro de um retângulo usando o método `createLinearGradient()`:

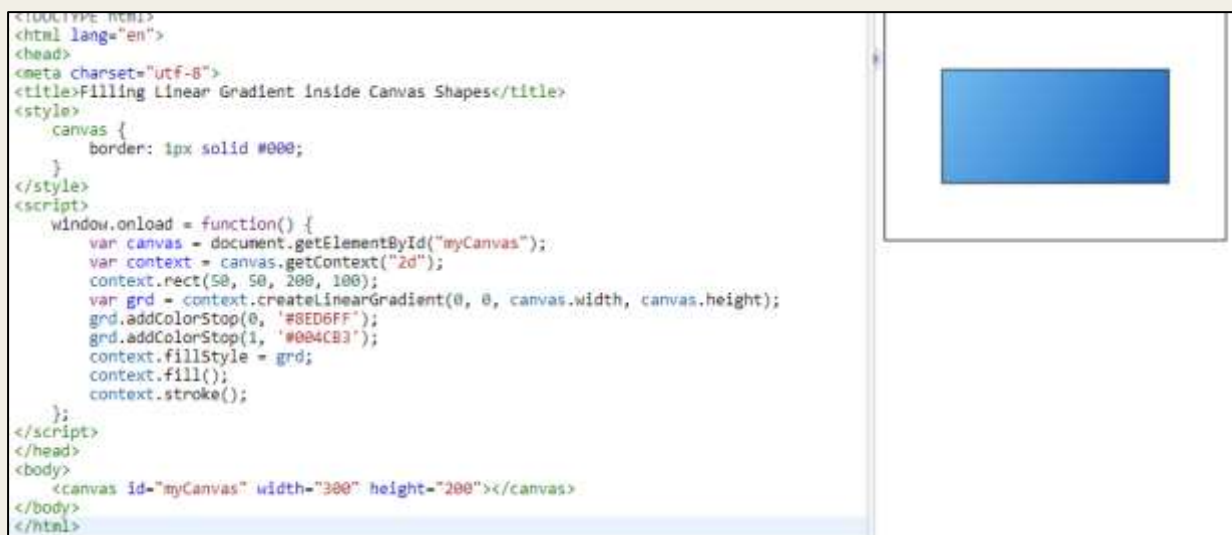


Figura 89 – Arquivar uma cor gradiente linear dentro de um retângulo usando o método `createLinearGradient()`

(Fonte: <https://www.tutorialrepublic.com/html-tutorial/html5-new-input-types.php>)

A *Figura 90* demonstra como preencher uma cor de gradiente radial dentro de um círculo por meio do método `createRadialGradient()`:

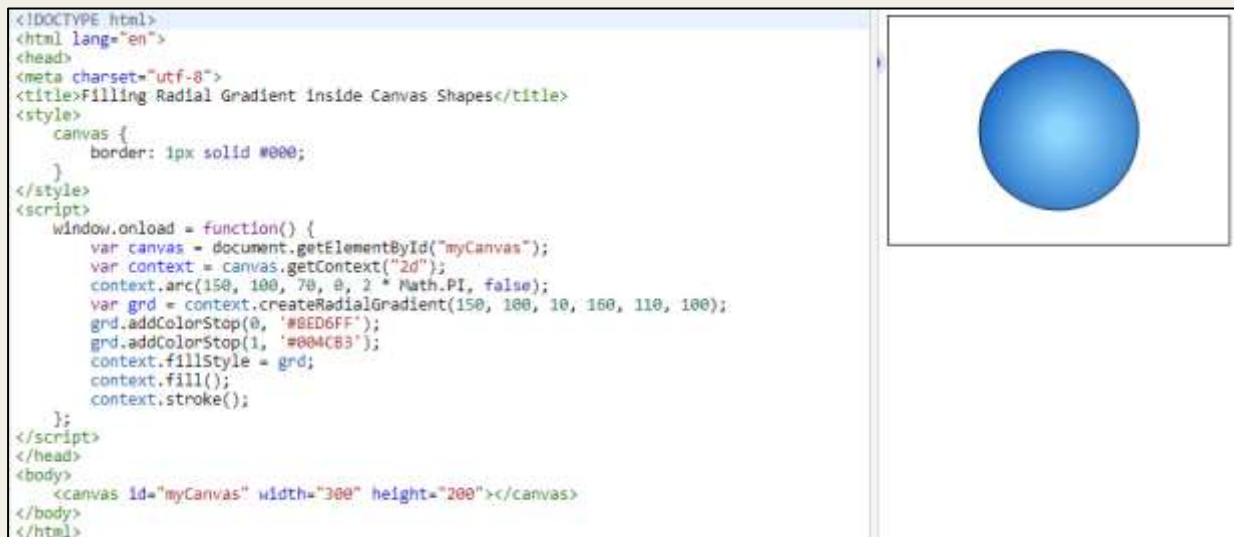


Figura 90 – Preencher uma cor gradiente radial dentro de um círculo através do método `createRadialGradient()`
(Fonte: <https://www.tutorialrepublic.com/html-tutorial/html5-new-input-types.php>)

Também é possível desenhar texto em tela (contendo apenas caracteres Unicode), além de **adicionar cor e alinhamento**, e até aplicar traço no texto usando o método `strokeText()`, que irá colorir o perímetro do texto em vez de preenchê-lo. No entanto, se o web designer pretende definir o preenchimento e o traço no elemento de texto, ele pode usar os métodos `fillText()` e `strokeText()` juntos. sugere-se usar o método `fillText()` antes do método `strokeText()` para renderizar o traço com precisão.



Figura 91 – Desenhar texto no canvas (Fonte: <https://www.tutorialrepublic.com/html-tutorial/html5-new-input-types.php>)

SVG HTML5

Espera-se que este subcapítulo seja claro sobre como usar elementos svg HTML5 para desenhar gráficos vetoriais numa página da web. Para isso, em primeiro lugar é importante definir SVG.

SVG significa Scalable Vector Graphics e é um formato de imagem baseado em XML que é usado para definir gráficos vetoriais bidimensionais para a web. Diferente da imagem rasterizada (por exemplo, .jpg, .gif, .png e outros formatos bidimensionais), uma imagem vetorial pode ser ampliada ou reduzida em qualquer medida sem perder a qualidade da imagem. As imagens vetoriais são compostas por uma série de formas definidas pela matemática, enquanto as imagens rasterizadas são compostas por um conjunto fixo de pontos (pixels). Como outros tópicos discutidos ao longo deste capítulo, o SVG também é uma recomendação do W3C.

Uma imagem SVG é construída usando uma sequência de instruções que acompanham o esquema XML, portanto, as imagens SVG podem ser criadas e editadas com um editor de texto, como o Bloco de Notas. Existem inúmeras vantagens para usar imagens SVG em vez de outros formatos de imagem, como segue:

- Elas podem ser pesquisadas, indexadas, roteirizadas e compactadas.
- Elas podem ser criadas e modificadas usando JavaScript em tempo real.
- Podem ser impressas com alta qualidade em qualquer resolução.
- Elas podem ser animadas usando os elementos de animação integrados.
- Podem conter hiperlinks para outros documentos.

Gráficos SVG podem ser incorporados diretamente num documento usando o elemento HTML5 `<svg>` (veja a *Figura 92* abaixo).

Todos os principais navegadores modernos (Chrome, Firefox, Safari e Opera), bem como o Internet Explorer 9 e superior, são compatíveis com renderização SVG inline.

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Embedding SVG Into HTML Pages</title>
<style>
  svg {
    border: 1px solid black;
  }
</style>
</head>
<body>
  <svg width="300" height="200">
    <text x="10" y="20" style="font-size:14px;">
      Your browser support SVG.
    </text>
    <text x="10" y="180" style="font-size:14px;">
      Sorry, your browser does not support SVG.
    </text>
  </svg>
</body>
</html>
```

Figura 92 – Incorporando gráficos SVG diretamente usando o elemento svg (Fonte: <https://www.tutorialrepublic.com/html-tutorial/html5-svg.php>)

Caminhos e formas básicas baseadas em vetor nas páginas da Web podem ser desenhados utilizando o elemento HTML5 `<svg>`.

O caminho mais básico para trabalhar com SVG é **desenhar uma linha reta**. Para que isso aconteça, o elemento SVG `<line>` deve ser usado. Como pode ser visto na Figura 93, os atributos `x1`, `x2`, `y1` e `y2` do elemento `<line>` desenharam uma linha de $(x1,y1)$ to $(x2,y2)$.

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Create a Line with HTML5 SVG</title>
<style>
  svg {
    border: 1px solid black;
  }
</style>
</head>
<body>
  <svg width="300" height="200">
    <line x1="50" y1="50" x2="250" y2="150" style="stroke:red; stroke-width:3;" />
  </svg>
</body>
</html>
```

Figura 93 – Desenhar uma linha reta com o elemento SVG `<line>` (Fonte: <https://www.tutorialrepublic.com/html-tutorial/html5-svg.php>)

Para desenhar um **retângulo e quadrados**, o elemento SVG `<rect>` é a forma mais adequada. Os atributos `x` e `y` do elemento `<rect>` especificam as coordenadas do canto

superior esquerdo do retângulo. Os atributos largura e altura especificam a largura e a altura da forma.



Figura 94 – Desenhar um retângulo com o elemento SVG <rect> (Fonte: <https://www.tutorialrepublic.com/html-tutorial/html5-svg.php>)

Se um **círculo** for a forma escolhida, o elemento SVG <circle> é o mais adequado. Os atributos cx e cy do elemento <circle> especificam as coordenadas do centro do círculo e o atributo r identifica o raio do círculo. Ainda assim, se os atributos cx e cy estiverem ausentes ou não especificados, o centro do círculo é definido como (0,0).

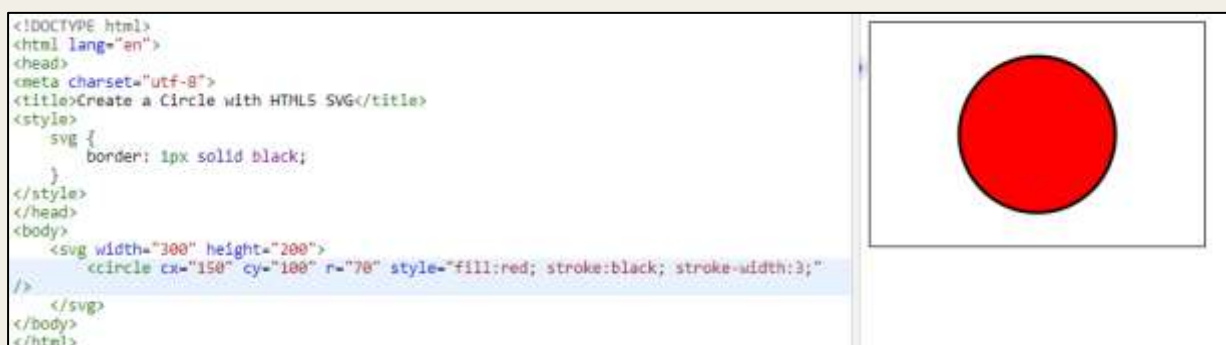


Figura 94 – Desenhar um círculo com o elemento SVG <circle> (Fonte: <https://www.tutorialrepublic.com/html-tutorial/html5-svg.php>)

Também é possível **desenhar texto** com SVG. O texto em SVG é renderizado como um gráfico para que o web designer possa usar toda a transformação gráfica nele, mas ainda funciona como texto, para que possa ser selecionado e copiado como texto pelo utilizador. Os atributos x e y do elemento <text> identificam a localização do canto

superior esquerdo em termos absolutos, embora os atributos dx e dy indiquem a localização relativa.



Figura 95 – Texto de desenho com elemento SVG <text> (Fonte: <https://www.tutorialrepublic.com/html-tutorial/html5-svg.php>)

Em alternativa, os web designers podem usar o elemento <tspan> para redimensionar ou realocar o intervalo de texto incluído num elemento <text>. O texto é incluído em **tspans** separados, mas dentro do mesmo elemento de texto podem ser seleccionados todos ao mesmo tempo — ao clicar e arrastar para seleccionar o texto. No entanto, o texto em elementos de texto separados não pode ser escolhido ao mesmo tempo.



Figura 96 – Desenho de texto com elemento SVG <tspan> (Fonte: <https://www.tutorialrepublic.com/html-tutorial/html5-svg.php>)

Embora os novos elementos gráficos `<canvas>` e `<svg>` tenham sido introduzidos pelo HTML 5 para criar gráficos de alta qualidade na web, eles diferem bastante.

Na tabela abaixo, as diferenças entre ambos são resumidas, e isso ajudará os alunos a usá-los de maneira apropriada e eficaz:

| SVG | Canvas |
|---|--|
| Baseado em vetor (composto de formas) | Baseado em rasterização (composto por pixels) |
| Vários elementos gráficos, que se tornam parte da árvore DOM da página | Elemento único semelhante a <code></code> no comportamento. O diagrama de canva pode ser guardado no formato PNG ou JPG |
| Modificado através de script e CSS | Modificado apenas por script |
| Bons recursos de renderização de texto | Más capacidades de renderização de texto |
| Dê melhor desempenho com menor número de objetos ou superfície maior, ou ambos | Dá melhor desempenho com grande número de objetos ou superfície menor, ou ambos |
| Melhor escalabilidade. Pode ser impresso com alta qualidade em qualquer resolução. A pixelização não ocorre | Má escalabilidade. Não é adequado para impressão em resolução mais alta. Pixelização pode ocorrer |

Figura 96 – As diferenças entre SVG e canvas (Fonte: <https://www.tutorialrepublic.com/html-tutorial/html5-svg.php>)

Áudio HTML 5

Este subcapítulo destina-se a explicar como incorporar áudio num documento HTML. Como os navegadores da web não tinham um padrão uniforme para incorporar arquivos de media como áudio, não era uma tarefa fácil de executar no passado. No entanto, existem muitas maneiras de incorporar som numa página da Web, desde simplesmente usar um link simples até usar o elemento HTML5 `<audio>`. Este elemento fornece uma maneira padrão de inserir áudio em páginas da web. Como o elemento de áudio é recente, ele é executado na maioria dos navegadores modernos.

Há muitas maneiras de inserir um áudio num documento HTML5. Uma delas é usar o grupo de controlos padrão do navegador, com uma fonte definida pelo atributo `src`, como pode ser verificado [neste código](#), em que é possível ouvir um grupo de pássaros a cantar.

Outra maneira pode ser alcançada usando o elemento `<object>`, que é usado para incorporar diferentes tipos de arquivos de media. [Este exemplo](#) inclui um arquivo de áudio numa página da web seguindo o método mencionado acima. Deve-se notar que o elemento `<object>` não é amplamente suportado e depende do tipo de objeto que está a ser implantado. Outros métodos, como o elemento HTML5 `<audio>` ou players de áudio HTML5 de terceiros podem ser uma opção melhor em muitos casos.

Finalmente, o elemento `<embed>` também pode ser outra forma de inserir media num documento HTML, seguindo [este exemplo](#). Embora o elemento `<embed>` seja muito bem suportado nos navegadores atuais e definido como padrão em HTML5, o áudio inserido pode não ser reproduzido devido à ausência de suporte do navegador para esse formato de arquivo ou inacessibilidade de plugins.

Vídeo HTML5

Agora é hora de aprender a incorporar conteúdo de vídeo num documento HTML.

Assim como o som, o conteúdo de vídeo também era difícil de inserir numa página da web, e pelo mesmo motivo (navegadores da web não tinham um padrão uniforme para definir arquivos de media incorporados como vídeo). Nos próximos parágrafos, muitas maneiras de inserir esses conteúdos serão explicadas.

O elemento `<video>` recém-introduzido funciona na maioria dos navegadores modernos. [Este exemplo](#) explica como simplesmente inserir um vídeo no documento HTML, usando o conjunto de controlos padrão do navegador, com uma fonte definida pelo atributo `src`. O elemento `<object>` também é usado para incorporar diferentes tipos de arquivos de media. Seguindo [este exemplo](#), pode-se entender como incorporar um vídeo Flash numa página da web (somente navegadores/aplicativos que suportem Flash poderão reproduzi-lo). Deve-se notar que o elemento `<object>` não é amplamente suportado e depende muito do tipo de objeto incorporado. Outros métodos podem ser uma escolha melhor em muitos casos, como, por exemplo, os dispositivos iPad e iPhone não podem exibir vídeos em Flash.

E quanto à **incorporação de vídeos do YouTube**? Essa é realmente a maneira mais simples e comum de incorporar arquivos de vídeo em páginas da web, hoje em dia. O web designer deve simplesmente fazer o upload do vídeo no YouTube e inserir o código HTML para exibir o vídeo na sua página da web. Aqui está um miniguia passo a passo:

Passo 1 – Carregar vídeo no YouTube.

Passo 2 – Após o upload de um vídeo no YouTube, o web designer deve procurar o botão 'partilhar', localizado abaixo de um vídeo em execução no player de vídeo da plataforma, como segue :



Figura 96 – Botão partilhar no YouTube (Fonte: Autor)

Ao clicar no botão 'partilhar', um painel para partilhar será aberto, exibindo mais algumas opções. Agora, o botão **'Incorporar'** deve ser clicado, pois ele irá gerar o código HTML para incorporar diretamente o vídeo na página da web. Para que isso aconteça, o web designer deve copiar e colar esse código no documento HTML.



Figura 97 – Opção "incorporar" no YouTube (Fonte: Autor)

Este código pode ser customizado ainda mais e para isso o web designer só precisa selecionar a opção de customização dada logo abaixo da caixa de entrada do código de incorporação.

A inserção de um vídeo do YouTube numa página da web é explicada [neste exemplo](#).

Armazenamento Web HTML5

Já se perguntou como usar o recurso de armazenamento na Web HTML5 para armazenar dados no navegador do utilizador? Os parágrafos a seguir serão úteis para entender isso completamente.

Em primeiro lugar, é importante entender as implicações do “armazenamento na web”.

Com o armazenamento na Web, os aplicativos da Web podem armazenar dados localmente, no navegador do utilizador. Antes do HTML5, os dados do aplicativo precisavam ser armazenados em cookies, incorporados em todas as solicitações do servidor. O armazenamento na Web é mais seguro e quantidades substanciais de dados podem ser armazenadas localmente, sem afetar o desempenho do site. As informações mantidas no armazenamento web não são enviadas ao servidor web, ao contrário dos cookies onde os dados são entregues ao servidor a cada solicitação. Além disso, os cookies permitem armazenar apenas uma pequena quantidade de dados (quase 4 KB), enquanto o armazenamento na web permite armazenar até 5 MB.

Existem dois tipos de armazenamento na web:

- **Armazenamento local** — usa o objeto `localStorage` para armazenar dados de todo o site de forma permanente. Dito isto, os dados locais armazenados estarão disponíveis no dia seguinte, na próxima semana ou no próximo ano, a menos que sejam removidos.
- **Armazenamento de sessão** — usa o objeto `sessionStorage` para armazenar dados temporariamente, para uma única janela ou guia do navegador. Os dados desaparecem quando a sessão termina, por exemplo, quando o utilizador fecha a janela ou guia do navegador.

No que diz respeito ao armazenamento local, cada dado é coletado num par chave/valor. A chave identifica o nome da informação (ou seja, 'first_name') e o valor é o valor relacionado à mesma chave (ou seja, 'Peter'). O [seguinte código JS](#) expressa o seguinte:

- `localStorage.setItem(chave, valor)` armazena o valor associado a uma chave.
- `localStorage.getItem(key)` salva o valor associado à chave.

Também é possível remover um item específico do armazenamento, passando o nome da chave para o método `removeItem()`, ou seja, `localStorage.removeItem("first_name")`.

No entanto, se o web designer pretende remover o armazenamento completo, ele deve usar o método `clear()`, ou seja, `localStorage.clear()`. O método `clear()` simplesmente limpa todos os pares de chave/valor do `localStorage` de uma só vez, portanto, **deve ser usado com cautela**. Os dados de armazenamento na web não estarão acessíveis entre diferentes navegadores.

Por fim, o objeto `sessionStorage` funciona de maneira semelhante ao `localStorage`, exceto que armazena os dados apenas para uma sessão. O [seguinte exemplo](#) é bastante explicativo em relação a como funciona.

Cache do Aplicativo HTML5

Durante o subcapítulo, os alunos podem aprender mais sobre como criar aplicativos offline usando o **recurso de cache HTML5**.

Sabe-se que a maioria dos aplicativos baseados na web não funcionará se o web designer estiver offline. No entanto, o HTML5 traz um mecanismo de cache de aplicativos que permite que o navegador guarde automaticamente o arquivo HTML e todos os outros recursos necessários para exibi-lo corretamente na máquina local, dessa forma o navegador ainda pode aceder à página da web e os seus recursos sem estabelecer uma conexão com a internet. Isso é suportado em todos os principais navegadores da Web modernos (Firefox, Chrome, Opera, Safari e Internet Explorer 10 e superior).

Existem várias vantagens em usar este recurso:

- **Navegação offline** — Os visitantes podem usar o aplicativo mesmo quando não estão online ou há interrupções inesperadas na conexão de rede.
- **Melhore o desempenho** — Os recursos armazenados em cache são carregados diretamente da máquina do utilizador, em vez do servidor remoto, para que as páginas da Web sejam carregadas mais rapidamente e tenham um desempenho melhor.
- **Reduzir a solicitação HTTP e a carga do servidor** — O navegador terá apenas que baixar os recursos atualizados/alterados do servidor remoto que reduzem as solicitações HTTP e economizam largura de banda valiosa, além de diminuir a carga no servidor web.

Existem algumas etapas a serem seguidas para armazenar em cache os arquivos para uso offline:

PASSO 1 – Crie um arquivo de manifesto de cache. Este é um arquivo de texto especial que informa aos navegadores quais os arquivos que eles devem armazenar (e não) e quais os arquivos que devem ser substituídos. Começa sempre com as palavras **CACHE MANIFEST** (sempre em maiúsculas).

A *Figura 98* abaixo é um exemplo de um arquivo de manifesto:

| Example | | Download |
|---------|---------------------|----------|
| 1 | CACHE MANIFEST | |
| 2 | # v1.0 : 10-08-2014 | |
| 3 | | |
| 4 | CACHE: | |
| 5 | # pages | |
| 6 | index.html | |
| 7 | | |
| 8 | # styles & scripts | |
| 9 | css/theme.css | |
| 10 | js/jquery.min.js | |
| 11 | js/default.js | |
| 12 | | |
| 13 | # images | |
| 14 | /favicon.ico | |
| 15 | images/logo.png | |
| 16 | | |
| 17 | NETWORK: | |
| 18 | login.php | |
| 19 | | |
| 20 | FALLBACK: | |
| 21 | /offline.html | |

Figura 98 – Exemplo de arquivo de manifesto (Fonte: <https://www.tutorialrepublic.com/html-tutorial/html5-application-cache.php>)

Agora é hora de explicar a codificação da Figura 98.

- Em primeiro lugar, é importante entender que os arquivos de manifesto podem ter três seções diferentes: **CACHE**, **NETWORK** e **FALLBACK**.
- Os arquivos listados sob o cabeçalho da seção **CACHE**: (ou logo após a linha **CACHE MANIFEST**) são claramente armazenados em cache após serem guardados pela primeira vez;
- Os arquivos sob o cabeçalho da seção **NETWORK**: são recursos da lista de permissões que nunca são armazenados em cache e estão disponíveis apenas online. Isso significa que os usuários nunca podem acessar a página **login.php** quando estiverem offline;
- A seção **FALLBACK**: especifica páginas alternativas que o navegador deve usar caso a conexão com o servidor não possa ser feita. Cada entrada nesta seção lista dois URLs — o primeiro é o recurso primário, o segundo é o fallback. Por exemplo, no caso da *Figura 98*, a página **offline.html** será exibida se o

utilizador estiver offline. Além disso, ambos os URLs devem ser da mesma origem que o arquivo de manifesto.

- Deve-se notar que as linhas que começam com '#' (símbolo de hash) são linhas de comentário.

Portanto, se houver um cache de aplicativo, o navegador carrega o documento e seus recursos associados diretamente do cache, sem aceder à rede. Em seguida, o navegador verifica se o arquivo de manifesto foi atualizado no servidor. Caso tenha sido atualizado, o navegador baixa a nova versão do manifesto e os recursos listados nele.

É importante observar que o próprio arquivo de manifesto não deve ser especificado no arquivo de manifesto de cache. Nesse caso, será muito difícil notificar o navegador de que um novo manifesto está disponível.

PASSO 2 – Use o arquivo de manifesto de cache. Depois de criá-lo, o web designer deve fazer upload do arquivo de manifesto de cache no servidor da web — certificando-se de que o servidor da web esteja configurado para servir os arquivos de manifesto com o tipo **MIME text/cache-manifest**.

Para fazer o cache manifest funcionar, o web designer precisará habilitá-lo nas páginas da web, adicionando o atributo manifest ao elemento raiz `<html>`, conforme mostrado na Figura 99 abaixo:

```
1 <!DOCTYPE html>
2 <html lang="en" manifest="example.appcache">
3 <head>
4   <title>Using the Application Cache</title>
5 </head>
6 <body>
7   <!--The document content will be inserted here-->
8 </body>
9 </html>
```

Figura 99 – Fazer o cache manifest funcionar (Fonte: <https://www.tutorialrepublic.com/html-tutorial/html5-application-cache.php>)

Se o utilizador está online, o resultado para este Código será o seguinte:

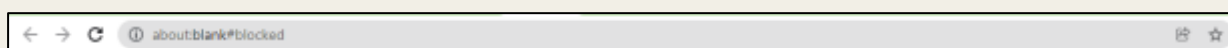


Figura 100 – about_blank#blocked (Fonte: <https://www.tutorialrepublic.com/html-tutorial/html5-application-cache.php>)

Trabalhadores da Web HTML5

Este subtópico será essencialmente útil para aprender mais sobre tópicos JS, pois ensinará como usar o web worker HTML5 para executar o código JS em segundo plano. Assim, os alunos devem voltar se tiverem alguma dificuldade.

Se alguém tentar realizar cálculos intensivos, demorados e pesados, exigindo tarefas com JavaScript, provavelmente irá congelar as páginas da Web e impedir que os utilizadores façam qualquer coisa até que o trabalho seja concluído. Por quê? Bem, porque o código JS é sempre executado em primeiro plano. No entanto, o HTML5 possui uma nova tecnologia ('web worker') criada para realizar o trabalho em segundo plano, além de outros scripts de interface do usuário, sem afetar o desempenho da página. Diferente das operações normais de JS, o web worker não interrompe o utilizador e a página da web permanece responsiva, porque estão as tarefas estão a ser executadas em segundo plano.

Os web workers são especialmente úteis para executar uma tarefa demorada. Assim, no primeiro exemplo, será criada uma tarefa JS simples que conta de zero a 100.000 (o nome do arquivo deve ser worker.js), conforme segue na *Figura 101*:

```
1 | var i = 0;
2 | function countNumbers() {
3 |     if(i < 100000) {
4 |         i = i + 1;
5 |         postMessage(i);
6 |     }
7 |
8 |     // Wait for sometime before running this script again
9 |     setTimeout("countNumbers()", 500);
10 | }
11 | countNumbers();
```

Figura 101 – Criar uma tarefa JS contando de 0 a 100.000 (Fonte: <https://www.tutorialrepublic.com/html-tutorial/html5-web-workers.php>)

Nota: Para uma melhor compreensão, é aconselhável baixar o código da Fig. 101 e seguir cada passo deste capítulo.

Desta forma, agora que o arquivo do web worker foi criado, é hora de iniciar o web worker a partir de um documento HTML. Esse documento executa o código dentro do arquivo chamado "worker.js", em segundo plano, e mostra gradualmente o resultado na página da web. Deve-se notar que o número à direita estará sempre a crescer, até atingir 100 000.

```

1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4 <meta charset="utf-8">
5 <title>Using HTML5 Web Workers</title>
6 <script>
7     if(window.Worker) {
8         // Create a new web worker
9         var worker = new Worker("/examples/js/worker.js");
10
11         // Fire onMessage event handler
12         worker.onmessage = function(event) {
13             document.getElementById("result").innerHTML = event.data;
14         };
15     } else {
16         alert("Sorry, your browser do not support web worker.");
17     }
18 </script>
19 </head>
20 <body>
21     <div id="result">
22         <!--Received messages will be inserted here-->
23     </div>
24 </body>
25 </html>

```

Figura 102 – Iniciar o web worker (Fonte: <https://www.tutorialrepublic.com/html-tutorial/html5-web-workers.php>)

Agora explicando o que está acontecendo no exemplo acima, a instrução **var worker = new Worker("worker.js");** cria um novo objeto no trabalhador da web, que é usado para comunicar com o trabalhador da web. Quando o trabalhador publica uma mensagem, ele aciona o manipulador de eventos **onmessage** (linha 14), que permite que o código receba mensagens do web worker. O elemento **event.data** compreende a mensagem enviada do **web worker**. Para o registo, o código que um trabalhador executa é sempre armazenado num arquivo JavaScript, separado para evitar que o desenvolvedor da Web escreva o código do trabalhador da Web, que tenta usar variáveis globais ou abrir diretamente os elementos na página da Web.

Também é possível **encerrar um worker em execução** no meio da operação, seguindo o exemplo abaixo:

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4 <meta charset="utf-8">
5 <title>Start/Stop Web Worker in HTML5</title>
6 <script>
7   // Set up global variable
8   var worker;
9
10  function startWorker() {
11    // Initialize web worker
12    worker = new Worker("/examples/js/worker.js");
13
14    // Run update function, when we get a message from worker
15    worker.onmessage = update;
16
17    // Tell worker to get started
18    worker.postMessage("start");
19  }
20
21  function update(event) {
22    // Update the page with current message from worker
23    document.getElementById("result").innerHTML = event.data;
24  }
25
26  function stopWorker() {
27    // Stop the worker
28    worker.terminate();
29  }
30 </script>
31 </head>
```



Figura 103 – Parar o running worker (Fonte: <https://www.tutorialrepublic.com/html-tutorial/html5-web-workers.php>)

O exemplo acima mostra como iniciar e parar o trabalhador de uma página da web simplesmente clicando nos botões HTML.

Eventos enviados pelo servidor HTML5

Este subcapítulo será útil para entender como usar o recurso de eventos enviados pelo servidor HTML5, para fazer uma conexão unidirecional e permanente entre uma página da Web e um servidor.

O evento enviado pelo servidor HTML5 é uma maneira inovadora de as páginas da Web comunicarem com um servidor da Web. No entanto, existem algumas circunstâncias em que as páginas da Web precisam de uma conexão de longo prazo com o servidor da Web, por exemplo, em cotações de ações em sites de finanças onde o preço é atualizado automaticamente ou tickers de jogos executados em vários sites de desportos. Tais coisas são viáveis com os eventos enviados pelo servidor HTML5, pois disponibiliza para uma página web manter uma conexão aberta com um servidor web, de forma que o servidor web possa enviar uma nova resposta mecanicamente a qualquer momento. Neste ponto, não há necessidade de reconectar e executar repetidamente o mesmo script de servidor desde o início.

Para uma melhor compreensão dos conceitos mencionados acima, use um arquivo PHP(1) chamado "server_time.php" e digite o seguinte script nele. Este arquivo irá apenas relatar a hora atual do relógio interno do servidor web em intervalos regulares:

```
1 <?php
2 header("Content-Type: text/event-stream");
3 header("Cache-Control: no-cache");
4
5 // Get the current time on server
6 $currentTime = date("h:i:s", time());
7
8 // Send it in a message
9 echo "data: " . $currentTime . "\n\n";
10 flush();
11 ?>
```

Figura 104 – exemplo server_time.php (Fonte <https://www.tutorialrepublic.com/html-tutorial/html5-server-sent-events.php>)

(1) Um arquivo PHP é um arquivo de texto simples que contém código escrito na linguagem de programação PHP. Como o PHP é uma linguagem de script do lado do servidor (back-end), o código escrito nele é executado no servidor. Na verdade,

um arquivo PHP pode conter texto simples, tags HTML ou código de acordo com a sintaxe PHP. O PHP é frequentemente usado para desenvolver aplicativos da Web que são processados por um mecanismo PHP no servidor da Web.

As duas primeiras linhas do script PHP (Fig. 104) definem dois cabeçalhos cruciais. Em primeiro lugar, define o tipo MIME para **text/event-stream**, que é necessário para o padrão de evento do lado do servidor. A segunda linha informa ao servidor web para desligar o cache ou então a saída do script pode ser armazenada em cache.

Cada mensagem enviada por meio de eventos enviados pelo servidor HTML5 deve começar com os dados de texto: seguido pelo texto da mensagem real e a nova sequência de caracteres de linha (**\n\n**).

E por último, a função PHP **flush()** foi usada para garantir que os dados são enviados imediatamente, em vez de armazenados em buffer até que o código PHP esteja completo.

Sobre como **processar mensagens em uma página web**, o objeto **EventSource** é utilizado para receber mensagens de eventos enviadas pelo servidor. No exemplo abaixo, os alunos verão como um documento HTML simplesmente recebe a hora atual informada pelo servidor da web e a exibe para os visitantes da página da web. Para um melhor entendimento, um documento HTML chamado **“demo_sse.html”** será criado e posteriormente colocado no mesmo diretório do projeto onde está localizado **“server_time.php”**. O download do código a seguir pode ser feito no link da fonte disponível abaixo.


```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4 <title>Using Server-Sent Events</title>
5 <script>
6     window.onload = function() {
7         var source = new EventSource("server_time.php");
8         source.onmessage = function(event) {
9             document.getElementById("result").innerHTML += "New time received
10 from web server: " + event.data + "<br>";
11         };
12     };
13 </script>
14 </head>
15 <body>
16     <div id="result">
17         <!--Server response will be inserted here-->
18     </div>
19 </body>
20 </html>
```

Figura 105 – Como processar mensagens numa página web (Fonte <https://www.tutorialrepublic.com/html-tutorial/html5-server-sent-events.php>)

Geolocalização HTML 5

Por meio deste subtópico, os alunos obterão algumas informações sobre como usar o recurso de geolocalização HTML5 para detetar a localização do utilizador. Esse recurso permite que o programador descubra as coordenadas geográficas (latitude e longitude) da localização atual do visitante do site. É especialmente útil para proporcionar a melhor experiência de navegação ao visitante, pois, por exemplo, esta ferramenta pode mostrar resultados de pesquisa fisicamente próximos da localização do utilizador.

Receber as informações de posição do visitante do site usando a API de geolocalização HTML5 não é difícil. Ele explora os três métodos incluídos no objeto `navigator.geolocation` — `getCurrentPosition()`, `watchPosition()` e `clearWatch()`.

Depois do utilizador concordar em permitir que o navegador informe ao servidor da Web sobre sua posição (os navegadores da Web não partilharão a localização do visitante

com uma página da Web, a menos que o usuário concorde em fazê-lo), o processo de geolocalização deve ocorrer da seguinte forma:

| | |
|--|--|
| <pre> <!DOCTYPE html> <html lang="en"> <head> <meta charset="utf-8"> <title>Get Visitor's Location Using HTML5 Geolocation</title> <script> function showPosition() { if(navigator.geolocation) { navigator.geolocation.getCurrentPosition(function(position) { var positionInfo = "Your current position is (" + "Latitude: " + position.coords.latitude + ", " + "Longitude: " + position.coords.longitude + ")"; document.getElementById("result").innerHTML = positionInfo; }); } else { alert("Sorry, your browser does not support HTML5 geolocation."); } } </script> </head> <body> <div id="result"> <!--Position information will be inserted here--> </div> <button type="button" onclick="showPosition();">Show Position</button> </body> </html> </pre> | <p>Your current position is (Latitude: 41.0079509, Longitude: -8.6270736)</p> <p>Show Position</p> |
|--|--|

Figura 106 – O processo de Geolocalização (Fonte: <https://www.tutorialrepublic.com/html-tutorial/html5-geolocation.php>)

Caso um utilizador não pretenda partilhar os seus dados de localização com o site, o programador pode fornecer duas funções, ao chamar a função `getCurrentLocation()`. A primeira função é chamada caso a tentativa de geolocalização seja bem-sucedida, enquanto a segunda é chamada se a tentativa de geolocalização falhar.

```

<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Handling the Geolocation Errors and Rejections</title>
<script>
// Set up global variable
var result;

function showPosition() {
// Store the element where the page displays the result
result = document.getElementById("result");

// If geolocation is available, try to get the visitor's position
if(navigator.geolocation) {
navigator.geolocation.getCurrentPosition(successCallback, errorCallback);
result.innerHTML = "Getting the position information...";
} else {
alert("Sorry, your browser does not support HTML5 geolocation.");
}
}

// Define callback function for successful attempt
function successCallback(position) {
result.innerHTML = "Your current position is (" + "Latitude: " + position.coords.latitude + ", " + "Longitude: " + position.coords.longitude + ")";
}

// Define callback function for failed attempt
function errorCallback(error) {
if(error.code == 1) {
result.innerHTML = "You've decided not to share your position, but it's OK. We won't ask you again.";
} else if(error.code == 2) {
result.innerHTML = "The network is down or the positioning service can't be reached.";
} else if(error.code == 3) {
result.innerHTML = "The attempt timed out before it could get the location data.";
} else {
result.innerHTML = "Geolocation failed due to unknown error.";
}
}
</script>
</head>
<body>
<div id="result">
<!--Position information will be inserted here-->
</div>
<button type="button" onclick="showPosition();">Show Position</button>
</body>
</html>

```

Figura 106 – Aplicando duas funções à função `getCurrentLocation()` (Fonte: <https://www.tutorialrepublic.com/html-tutorial/html5-geolocation.php>)

Existem muitas funções interessantes para explorar com dados de geolocalização, como mostrar a localização do usuário no Google Maps. Com base nos dados de latitude e longitude recuperados por meio do recurso de geolocalização HTML5, [este exemplo](#) mostra a localização atual do leitor. Isso simplesmente mostra uma imagem estática mostrando a localização do usuário, embora um Google Maps interativo com arrastar, aumentar/diminuir o zoom e outros recursos, como [este exemplo mostra](#).

Todos os exemplos mencionados acima foram baseados no método `getCurrentPosition()`. No entanto, a função de geolocalização possui outra técnica – `watchPosition()` que permite rastrear o movimento do visitante retornando a posição atualizada à medida que a localização muda. `watchPosition()` tem os mesmos parâmetros de entrada que `getCurrentPosition()`. No entanto, `watchPosition()` pode

ativar a função de sucesso várias vezes - quando obtém a localização pela primeira vez e, novamente, toda vez que identifica uma nova posição, como [este exemplo sugere](#).

Arrastar e soltar HTML5

É um procedimento comum na rotina diária online, arrastar e soltar um elemento para outro local em um site. O recurso HTML5 Drag and Drop permite fazer isso, e qualquer elemento pode ser arrastado e solto. Este [exemplo w3schools](#) define um exemplo simples de arrastar e soltar, os alunos podem tentar se familiarizar mais com esse conceito. Embora o código pareça difícil de entender, é bastante simples e lógico:

- Em primeiro lugar, para tornar um elemento arrastável, o atributo arrastável deve ser verdadeiro:

```
<img draggable="true">
```

- Em seguida, deve-se especificar o que deve acontecer quando o elemento for arrastado. No exemplo w3schools dado acima, o atributo **ondragstart** chama uma função (**arrastar (evento)**) que especifica quais dados serão arrastados. O processo **dataTransfer.setData()** define o tipo de dados e o valor dos dados arrastados:

```
function drag(ev)  
{ev.dataTransfer.setData("text",ev.target.id);}
```

Neste exemplo, o tipo de dado é "texto", sendo o valor o id do elemento arrastável ("drag1").

- O evento **ondragover** estipula onde os dados arrastados podem ser soltos. Por padrão, os dados/elementos não podem ser descartados noutros elementos. Para permitir uma queda, o web designer deve evitar a manipulação padrão do elemento, chamando a técnica **event.preventDefault()** para o evento **ondragover**:

```
event.preventDefault()
```

- Assim que os dados arrastados são soltos, ocorre um evento de soltar. No exemplo dado anteriormente, o atributo ondrop chama uma função, `drop(event)`:

```
function drop(ev) {  
  ev.preventDefault();  
  var data=ev.dataTransfer.getData("text");  
  ev.target.appendChild(document.getElementById(data));  
}
```

- ✓ Chame `preventDefault()` para evitar que o navegador manipule os dados por padrão (o padrão é aberto como um link ao soltar);
- ✓ O programador obtém os dados arrastados com a técnica `dataTransfer.getData()`. Essa técnica retornará todos os dados que foram configurados para o mesmo tipo na técnica `setData()`;
- ✓ Os dados arrastados são o id do elemento arrastado ("`drag1`")
- ✓ O programador também deve anexar o elemento arrastado no elemento soltar.

Referências HTML5

Para obter uma lista detalhada e abrangente de elementos relacionados a **tags/elementos HTML5**, **atributos globais HTML5**, **atributos de evento HTML5**, **códigos de idioma HTML5**, **entidades de caracteres HTML5**, **códigos de status HTTP**, **seletor de cores HTML5** e outras referências úteis, os alunos devem consultar o [seguinte link](#) (HTML5 References section).