Cábula de HTML e CSS · Criar páginas web

Bases · Texto · Ligações · Semântica · Imagens · Ajuda

Holger Sebastião · www.sebastiao.org
Versão 1.1 · 2026-10-25

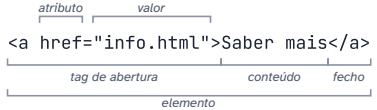
Estrutura base — index.html

```
<!DOCTYPE html>
<html lang="pt">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content=
    "width=device-width, initial-scale=1">
  <title>A minha página</title>
  <link rel="stylesheet"
        href="css/style.css">
</head>
<body>
  <h1>Olá mundo</h1>
</body>
</html>
```

`<!DOCTYPE html>`	Isto é HTML5. Sempre a primeira linha.
`<head>`	Informação para o browser, invisível: carateres, título do separador, CSS.
`<body>`	Tudo o que se vê na página.

A página inicial chama-se `index.html`.

Estrutura de um elemento



A tag de fecho tem uma `/`. Os atributos ficam na tag de abertura: `nome="valor"`, o valor entre aspas.

Elementos vazios não têm conteúdo nem tag de fecho: `<br>` `<hr>` `<img>` `<meta>` `<link>`

Encaixar bem, fechar primeiro o de dentro: `<p><strong>…</strong></p>`, não `<p><strong>…</p></strong>`

Escreve as tags em minúsculas e usa a indentação para mostrar o que está dentro de quê.

Carateres especiais

`<`	<	`>`	>
`&`	&	` `	espaço sem quebra de linha
`©`	©		

Acentos e cedilha (á, ã, ç) escrevem-se diretamente, graças a `charset="UTF-8"`.

Texto

```
<h1>Título principal</h1>
<h2>Subtítulo</h2>   <!-- até há -->

<p>Um parágrafo. Aqui fica texto que
no browser simplesmente continua.</p>

<p>Primeira linha<br>
segunda linha no mesmo parágrafo</p>

<strong>importante</strong>  <em>ênfase</em>
<hr>   <!-- linha divisória -->
```

`h1` … `h6`	títulos. `h1` só uma vez por página, não saltar níveis.
`p`	parágrafo (paragraph)
`br`	quebra de linha dentro de um parágrafo
`strong`, `em`	importante (negrito), ênfase (itálico)

O browser ignora os espaços e as mudanças de linha do código. Não uses `<br>` para espaçar, para isso há CSS.

Comentário: `<!-- não aparece -->`

Listas

```
<ul>
  <li>Leite</li>
  <li>Pão</li>
</ul>
```

```
<ol>
  <li>Acordar</li>
  <li>Duche</li>
</ol>
```

`ul`	lista com marcas (unordered list)
`ol`	lista numerada (ordered list)
`li`	um item da lista (list item), só em `ul` ou `ol`

Encaixar: a lista interior fica **dentro** do `<li>` da exterior.

Tabela

```
<table>
  <tr><th>Nome</th><th>Nota</th></tr>
  <tr><td>Ada</td><td>18</td></tr>
  <tr><td>Max</td><td>15</td></tr>
</table>
```

`tr` linha (table row), `th` célula de cabeçalho, `td` célula de dados. Tabelas só para dados, nunca para o layout da página.

Ligações — a href

```
<a href="contacto.html">Contacto</a>
```

DESTINO	HREF
página na mesma pasta	`contacto.html`
página numa subpasta	`paginas/equipa.html`
um nível acima	`../index.html`
outro site	`https://visucode.de`
sítio na página	`#contacto`
e-mail	`mailto:info@escola.pt`

```
<!-- abrir num separador novo -->
<a href="https://pt.wikipedia.org"
   target="_blank">Wikipédia</a>

<!-- destino do salto: id sem # -->
<h2 id="contacto">Contacto</h2>

<!-- imagem como ligação -->
<a href="index.html"><img src="logo.png"
   alt="Para a página inicial"></a>
```

Caminho relativo (`paginas/equipa.html`): a partir da pasta do ficheiro atual, também funciona no servidor. Um **caminho absoluto** começa com `https://`.

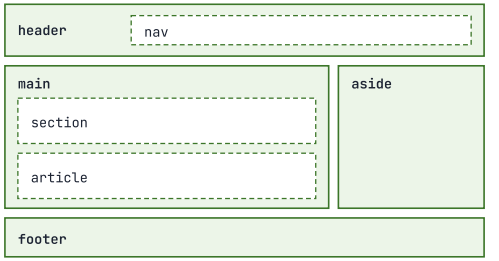
Nunca `C:\Users\…` nem `/Users/…` como destino: isso só existe no teu computador.

Navegação — nav

```
<header>
  <nav>
    <ul>
      <li><a href="index.html">Home</a></li>
      <li><a href="turma.html">Turma</a></li>
      <li><a href="info.html">Info</a></li>
    </ul>
  </nav>
</header>
```

Um menu é uma **lista de ligações** dentro de `<nav>`. O mesmo menu aparece em todas as páginas. O CSS transforma-o numa barra (página 2).

HTML5 semântico — estrutura da página



`header`	topo: logótipo, título, menu
`nav`	navegação principal
`main`	conteúdo principal, só uma vez por página
`section`	secção com título próprio
`article`	vale por si, por exemplo um artigo
`aside`	informação lateral, por exemplo uma caixa
`footer`	rodapé: ficha técnica, contacto

Semântico quer dizer: a tag diz **o que** é o conteúdo, não como aparece. Ajuda os leitores de ecrã e os motores de busca. `div` e `span` só quando nenhuma tag serve (por exemplo, só para CSS).

Imagens — img

```
<figure>
  <img src="imagens/cao.jpg"
       alt="Cão castanho na relva"
       width="300">
  <figcaption>O cão da escola</figcaption>
</figure>
```

`alt` é obrigatório: descreve a imagem para pessoas cegas e quando não carrega. `figure` + `figcaption` = imagem com legenda.

Erros típicos

- Tag de fecho esquecida ou mal encaixada
- Nomes de ficheiros com espaços, acentos ou maiúsculas: melhor `sobre-nos.html`. O servidor distingue `Foto.JPG` de `foto.jpg`.
- Caminho errado: o ficheiro está mesmo nessa pasta?
- Verificar: `validator.w3.org`, no browser F12 ou botão direito → Inspecionar.

Cábula de HTML e CSS · Criar páginas web

Incluir · Seletores · Estilo · Caixa · Layout · Ajuda

Holger Sebastião · www.sebastiao.org
Versão 1.1 · 2026-10-25

Incluir CSS

3 formas

1 · EXTERNO: É ASSIM QUE FAZEMOS

```
<!-- no head de cada página HTML -->
<link rel="stylesheet" href="css/style.css">
```

```
/* ficheiro css/style.css: só CSS */
body {
  font-family: Arial, sans-serif;
}
```

Um ficheiro para todas as páginas: se o mudares, mudam todas.

2 · INTERNO NO HEAD

```
<head>
  <style>
    h1 { color: navy; }
  </style>
</head>
```

Só vale para esta página.

3 · DIRETAMENTE NA TAG (INLINE)

```
<p style="color: red;">Atenção</p>
```

Só vale para este elemento. Só para testes rápidos.

> **Quem ganha?** Inline ganha a interno e externo. De resto, ganha o que vem depois.

Estrutura de uma regra



O **seletor** escolhe **o que** recebe estilo. Dentro de `{ }` ficam as declarações, cada uma termina com `;`. Comentário: `/* … */`

Herança em CSS

```
body {
  font-family: Arial, sans-serif;
  color: #1d2433;
}
```

O tipo de letra e a cor do texto de `body` valem para tudo o que está dentro. Espaços, contornos e fundos **não** se herdam.

Seletores

CSS	APANHA	NO HTML
`p`	todos os parágrafos	`<p>`
`.aviso`	classe, quantas vezes quiseres	`class="aviso"`
`#topo`	id, só uma vez por página	`id="topo"`
`nav a`	ligações **dentro** de nav	
`h1, h2`	h1 e h2	
`a:hover`	ligação debaixo do rato	
`*`	todos os elementos	

```
<p class="aviso">Amanhã não há aulas</p>
```

```
.aviso {
  background-color: #fff3c4;
}
```

Se várias regras servirem: id ganha a classe, classe ganha a elemento. Em caso de empate, ganha a regra que vem depois.

Cores e texto

```
h1 {
  color: #1f5fae;       /* cor do texto */
  background-color: white;
  font-family: Arial, sans-serif;
  font-size: 2rem;
  font-weight: bold;
  text-align: center;   /* ou left */
  line-height: 1.5;     /* entrelinha */
}
a {
  text-decoration: none; /* sem traço */
}
```

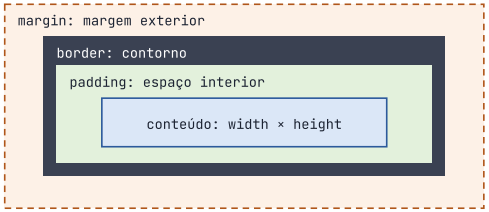
cor	`red` · `#ff0000` · `rgb(255, 0, 0)`
transparente	`rgba(0, 0, 0, 0.5)`
unidades	`px` fixo · `%` do elemento pai · `rem` do texto base (16px)

Dar estilo às ligações

```
a:link    { color: #1f5fae; } /* nova */
a:visited { color: #6b4aa6; } /* visitada */
a:hover   { color: #b0561a; } /* rato */
a:active  { color: #b3261e; } /* clique */
```

Respeita esta ordem, senão reescrevem-se umas às outras. Para memorizar: **LoVe HAte.**

Modelo de caixa




```
.cartao {
  width: 300px;
  padding: 1rem;
  border: 2px solid #3a4152;
  border-radius: 8px;  /* cantos redondos */
  margin: 20px auto;   /* auto = ao centro */
  box-sizing: border-box;
}
```

`box-sizing: border-box`: o `width` inclui padding e border. Forma curta `margin: 10px 20px;` = cima/baixo 10, esquerda/direita 20. Quatro valores: cima, direita, baixo, esquerda (sentido dos ponteiros).

Navegação em barra

Flexbox

```
nav ul {
  list-style: none;   /* sem marcas */
  margin: 0;
  padding: 0;
  display: flex;      /* lado a lado */
  gap: 1rem;
}
nav a {
  color: white;
  text-decoration: none;
  padding: 0.5rem 1rem;
}
nav a:hover {
  background-color: #1f5fae;
}
```

`display: flex` põe os filhos lado a lado. Também: `justify-content: center` (horizontal), `align-items: center` (vertical), `flex-direction: column` (uns por baixo dos outros).

Telemóvel e tablet

@media

```
/* só vale até 600px de largura */
@media (max-width: 600px) {
  nav ul {
    flex-direction: column;
  }
}
```

Precisa da linha viewport no head (estrutura base).

Erros típicos

- Falta `;` ou `}`: daí em diante nada funciona.
- No CSS `.aviso` com ponto, no HTML `class="aviso"` sem.
- Caminho errado no `<link>`: `style.css` está em `css`?
- Versão antiga: recarregar com Ctrl+F5 ou Cmd+Shift+R.
- Que regra se aplica? F12 ou botão direito → Inspecionar.

Pastas de um site

```
website/
├── index.html
├── turma.html
├── contacto.html
├── css/
│   └── style.css
└── imagens/
    └── logo.png
```

A partir de `index.html`: `css/style.css`, `imagens/logo.png`. Tudo em minúsculas, sem espaços.

Layout da página

Flexbox

```
<div class="conteudo">
  <main>…</main>
  <aside>…</aside>
</div>
```

```
.conteudo {
  display: flex;
  gap: 2rem;
}
main  { flex: 3; }   /* 3/4 da largura */
aside { flex: 1; }   /* 1/4 da largura */
```

Cábula de HTML e CSS · Criar páginas web

Estrutura · Campos · Atributos · Escolhas · CSS · Ajuda

Holger Sebastião · www.sebastiao.org
Versão 1.1 · 2026-10-25

Estrutura de um formulário `form`

```
<form action="enviar.php" method="post">
  <label for="nome">Nome</label>
  <input type="text" id="nome"
         name="nome" required>

  <button type="submit">Enviar</button>
</form>
```

`action`	para onde vão os dados (ficheiro no servidor)
`method`	`get`: os dados aparecem na barra de endereço · `post`: invisível, para palavras-passe e muitos dados
`name`	o valor chega ao servidor com este nome
`id`	liga o campo ao seu `label`

label for = id: clicar no texto põe o cursor no campo. Os leitores de ecrã leem o texto em voz alta.

Sem `name`, o campo **não** é enviado.

O que acontece ao enviar?



O HTML só mostra o formulário. Só um programa no servidor pode guardar os dados, por exemplo em PHP com SQL.

Botões `button`

```
<button type="submit">Enviar</button>
<button type="reset">Limpar</button>
<button type="button">Só botão</button>
```

`submit` envia, `reset` limpa todos os campos, `button` não faz nada (para JavaScript). Indica sempre o `type`.

Testar sem servidor

```
<form action="obrigado.html" method="get">
```

Cria uma página vazia `obrigado.html`. Depois de enviar, a barra de endereço mostra:
`obrigado.html?nome=Ali`
Assim vês o que o formulário envia.

Campos de entrada `input`

TYPE	PARA
`text`	texto curto, uma linha
`email`	e-mail, verifica o @
`password`	só mostra pontos
`number`	número, com `min` `max` `step`
`date`	data com calendário
`tel`	número de telefone
`url`	endereço web, verifica https://
`color` · `range`	seletor de cor · barra deslizante
`file`	carregar um ficheiro
`checkbox`	caixa de seleção, várias possíveis
`radio`	exatamente um de um grupo
`submit`	botão de enviar (melhor `<button>`)

```
<label for="idade">Idade</label>
<input type="number" id="idade"
       name="idade" min="14" max="99">

<label for="mail">E-mail</label>
<input type="email" id="mail" name="mail"
       placeholder="nome@escola.pt">
```

Atributos importantes

`required`	obrigatório, senão não envia
`placeholder`	texto de exemplo cinzento no campo vazio
`value`	valor enviado (ou valor inicial)
`maxlength="20"`	no máximo 20 carateres
`min` · `max`	menor e maior número ou data
`checked`	visto ou ponto já marcado
`disabled`	acinzentado, não é enviado
`readonly`	só de leitura, é enviado
`autofocus`	o cursor fica no campo ao carregar

O placeholder não substitui o label: o texto desaparece ao escrever.

Escolher `radio` · `checkbox`

```
<fieldset>
  <legend>Ano</legend>
  <input type="radio" id="a11"
         name="ano" value="11" checked>
  <label for="a11">11.º</label>
  <input type="radio" id="a12"
         name="ano" value="12">
  <label for="a12">12.º</label>
</fieldset>

<input type="checkbox" id="termos"
       name="termos" required>
<label for="termos">Concordo</label>
```

radio: mesmo `name` = um grupo, só se escolhe um. Cada um precisa do seu `value`.
`fieldset` agrupa campos relacionados, `legend` é o título.

Lista e texto longo

```
<label for="disc">Disciplina</label>
<select id="disc" name="disc">
  <option value="">Escolhe</option>
  <option value="inf">Informática</option>
  <option value="mat">Matemática</option>
</select>

<label for="texto">Mensagem</label>
<textarea id="texto" name="texto"
          rows="4"></textarea>
```

A `textarea` tem tag de fecho e não tem `value`: o texto inicial fica entre as tags.

O que chega ao servidor?

CAMPO	É ENVIADO
campo de texto	`nome=Ali`
radio	`ano=12` (o `value` escolhido)
checkbox ligada	`termos=on` (sem `value`)
checkbox desligada	nada
select	`disc=inf` (o `value`, não o texto)

Com `get`, os pares aparecem na barra de endereço unidos por `&`: `?nome=Ali&ano=12`

Dar estilo a formulários `CSS`

```
label {
  display: block;      /* linha própria */
  margin-top: 0.8rem;
}
input, select, textarea {
  width: 100%;
  padding: 0.4rem;
  border: 1px solid #8a92a3;
  border-radius: 4px;
  box-sizing: border-box;
}
input:focus, textarea:focus {
  outline: 2px solid #1f5fae; /* ativo */
}
input:invalid {
  border-color: #b3261e; /* erro */
}
button {
  margin-top: 1rem;
  padding: 0.5rem 1.5rem;
}
```

Em radio e checkbox, deixar `width: auto` e pôr o `label` ao lado com `display: inline`.

```
input[type="radio"],
input[type="checkbox"] { width: auto; }
```

label sem for

```
<label>
  <input type="checkbox" name="news">
  Subscrever a newsletter
</label>
```

Se o campo estiver **dentro** do `label`, não precisa de `for` nem de `id`. Prático em checkbox e radio.

Erros típicos

- Falta o `name`: o valor não chega.
- `for` e `id` escritos de forma diferente.
- Botões radio com `name` diferente: dá para escolher todos.
- `<input>` não tem tag de fecho, `<textarea>` e `<select>` têm.
- Botão fora do `<form>`: não envia nada.
- `required` não substitui a verificação no servidor.