HTML

Written by Febogamedev

HTML, an acronym for **HyperText Markup Language**, is the fundamental m**arkup language** used to describe the structure and meaning of the content of a Web page (WHATWG, 2026c).
A file containing HTML code has the .html extension.

A markup language is a system that uses elements and markers, commonly called **tags**, to describe the structure and meaning of the different parts of a document (WHATWG, 2026c).
A markup language does not primarily specify a sequence of operations to execute, as a programming language does, but describes the role and organisation of content (WHATWG, 2026c).

An *HTML* document consists of elements, which are the structures represented by tags (WHATWG, 2026c).
An *HTML* element consists of:

- The opening tag: identifies the type of element.
- Attribute-value pairs: Some elements often need attributes to specify certain parameters.
- Content: The actual content of the element.
- Closing tag: indicates the end of the element.

HTML elements form a hierarchical structure (WHATWG, 2026a, 2026c).
Example:

```
<main>
  <article>
    <h1>Heading</h1>
    <p>Article content.</p>
  </article>
</main>
```

We can represent it as a tree:

```
main
└── article
    ├── h1
    └── p
```

This hierarchy communicates the relationships between the contents.
The logical structure does more than arrange elements on the screen: it helps browsers, search engines and assistive technologies understand the document (WHATWG, 2026c; World Wide Web Consortium [W3C], 2026b).

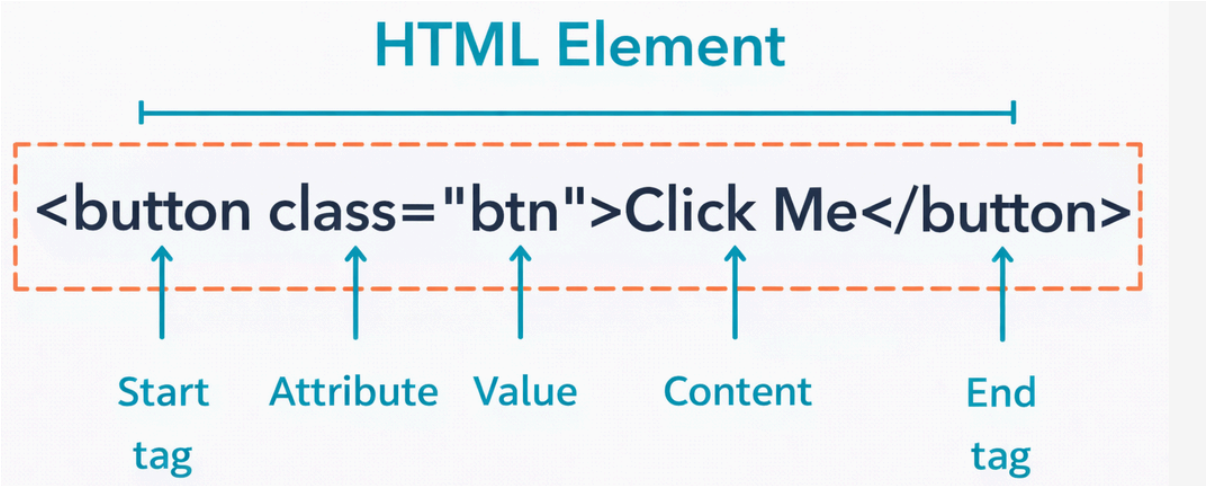
Headings, for example, also allow assistive technologies to offer tools for navigating between sections of the page (W3C, 2024b, 2026b).

Tags

Tags are language elements enclosed in angle brackets (< >), a portion of markup that delimits or identifies an element (WHATWG, 2026c).
This structuring allows browsers and other software agents to correctly interpret the meaning of content and therefore display it correctly (WHATWG, 2026c; W3C, 2026b).
For example:

- The <h1> tag is used for the *HTML* element that defines the main heading of a page (WHATWG, 2026c).
- The <p> tag is used to define paragraphs of text. The <a> tag is used to create hyperlinks (WHATWG, 2026c).

Tag and element are often used as synonyms, but "tag" refers to the specific part of the code enclosed in angle brackets, while "element" refers to the entire object, including the opening tag, the content and the closing tag (WHATWG, 2026c).



Example

```
<a href="https://example.com">Visit the site</a>
```

- : opening tag.
- href="https://example.com": attribute and value.
- Visit the site: content.
- : closing tag.

The whole set is the HTML element.

Note:

The tag name “a” comes from anchor and identifies an element that can create a hyperlink.

Many elements can contain text and other elements:

```
<p>This is an <strong> important text </strong> </p>
```

In this example:

- .. represents the paragraph element.
- ... is the element contained within it: this element is therefore a child of the first.

Elements must be nested correctly. The outer element must not be closed before the inner one (WHATWG, 2026c).

- Correct: `<p> This is an <strong> important text </strong> </p>`
- Incorrect: `<p> This is an  <strong> important text </p> </strong>`

Some elements cannot contain text or other elements (WHATWG, 2026c).
They are called void elements because they have no content and only have an opening tag (WHATWG, 2026c).
An example is <img>:

```
<img src="paesaggio.jpg" alt="A valley surrounded by mountains">
```

The element does not have an </img> tag (WHATWG, 2026c).
An *HTML* void element consists of:

- the opening tag: identifies the type of element.
- Attributes and their values: Some elements often need attributes to specify certain parameters.

Attributes add information or settings to an element (WHATWG, 2026c).
They are inserted in the opening tag: <element-name attribute="value">
Example:

```
<img src="paesaggio.jpg" alt="A valley surrounded by mountains">
```

- “src” indicates the address of the graphical resource;
- “alt” provides the image’s text alternative (WHATWG, 2026c; W3C, 2024b).

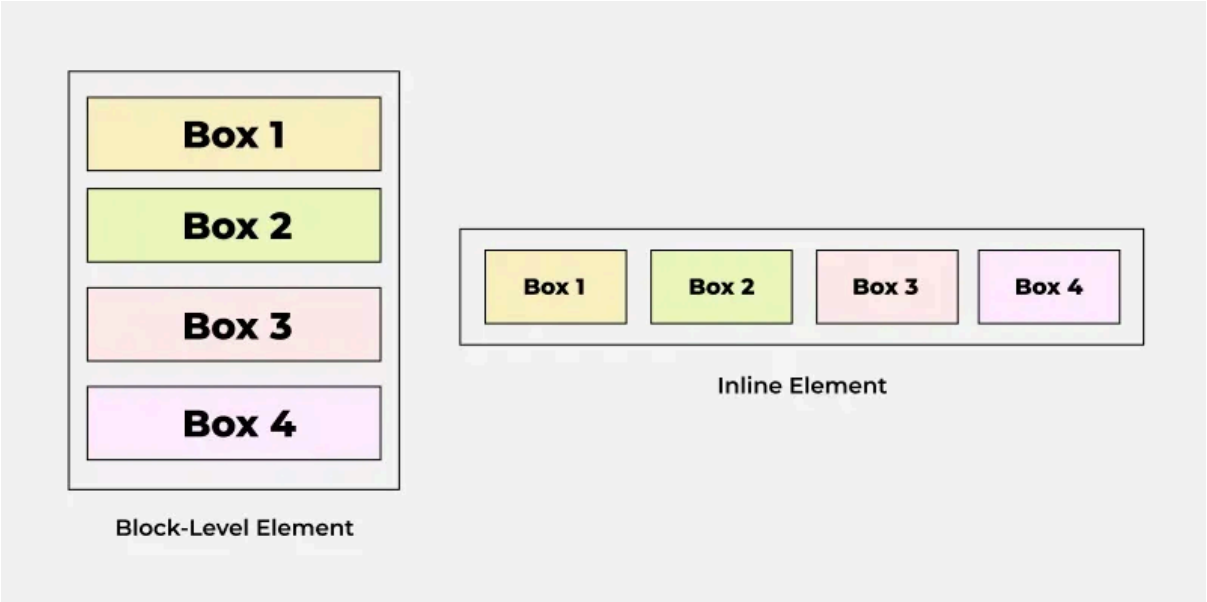
Some attributes are available for almost all elements, such as the “id” attribute (WHATWG, 2026c).
Others belong only to certain elements.
For example, “href” is used in links and “src” in images or other embedded resources (WHATWG, 2026c).
The meaning of the attribute therefore depends on the element in which it is used.

In HTML, elements can have two main behaviours in page layout: **block** and **inline (W3C, 2026a; WHATWG, 2026c)**.

Block elements normally start on a new line and occupy the full available width of their container, even when their content would require less space. The elements that follow are therefore placed below them (W3C, 2026a; WHATWG, 2026c).
Elements such as , <p>, <h1> and <section> are block by default (WHATWG, 2026c).

Inline elements, on the other hand, do not start on a new line and occupy only the space required by their content (W3C, 2026a; WHATWG, 2026c).
They can therefore be placed next to each other on the same line, like the words in a sentence. Elements such as , <a>, and are inline by default (WHATWG, 2026c).

This distinction describes the default visual behaviour assigned by the browser, rather than an immutable characteristic of the *HTML* element: *CSS* can be used to change this behaviour (W3C, 2026a).



Structure of an HTML document

A modern *HTML* document can start from this structure (WHATWG, 2026c):

```
<!doctype html>
<html lang="it">

<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>Example HTML page</title>
</head>
```

```
<body>
  <header>
    <h1>Welcome to my page</h1>
  </header>

  <main>
    <p>This is a simple HTML document.</p>

    <img
      src="paesaggio.jpg"
      alt="A valley surrounded by mountains">
  </main>
</body>

</html>
```

Let us analyse it element by element:

DOCTYPE

```
<!doctype html>
```

The DOCTYPE is not an *HTML* element and does not have a closing tag (WHATWG, 2026c).
It ensures that the browser uses the modern standards-compliant mode, rather than historical modes created to interpret older pages (WHATWG, 2026c).

LANG

```
<html lang="it">
```

html is the root element of the document, and the lang attribute with the value it indicates that the main language is Italian (WHATWG, 2026c).
This information helps, among others, speech synthesis systems and translation tools to interpret the text correctly (WHATWG, 2026c; W3C, 2024b).

HEAD

```
<head>
 ...
</head>
```

The head section contains information that describes or configures the document (WHATWG, 2026c).
In head we can find:

- Document title.
- Character encoding.
- Metadata.
- Links to stylesheets.
- Icons.

- Scripts.
- Information for browsers and search engines.

CHARSET

```
<meta charset="utf-8">
```

indicates that the document uses *UTF-8* as its character encoding (WHATWG, 2026b, 2026c).
The HTML standard requires utf-8 as the value of the charset declaration (WHATWG, 2026b, 2026c).

VIEWPORT

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

Helps the mobile browser use a **viewport** consistent with the device's width (W3C, 2024a).

The viewport is the rectangular area through which the browser displays a Web page. It does not necessarily coincide with the entire page: if the document is longer or wider than the viewport, the user can scroll to see its other parts (W3C, 2024a).

On a computer, the viewport normally corresponds to the inner area of the browser window, excluding elements such as tabs and the address bar. Its dimensions change when the window is resized (W3C, 2024a).

The two parameters mean that:
- "width=device-width" sets the width of the viewport in relation to the width available on the device (W3C, 2024a).
- "initial-scale=1" sets the initial zoom level of the page (W3C, 2024a).

This declaration is particularly important on mobile devices. Without it, the browser might simulate a wider viewport, similar to that of a computer, and then shrink the entire page, making text and components appear very small (W3C, 2024a).

TITLE

```
<title>Example HTML page</title>
```

defines the title of the document (WHATWG, 2026c).
The browser can use it to identify the page in the tab, window, history or bookmarks (WHATWG, 2026c).
It does not necessarily coincide with the main heading visible on the page, which is normally expressed with <h1> (WHATWG, 2026c).
The standard explicitly distinguishes the document title from the content headings (WHATWG, 2026c).

BODY

```
<body>
...
```

```
</body>
```
contains the main structure of the document and the elements with which the user interacts (WHATWG, 2026c).
Not all elements in body necessarily have to be visible at all times: some may be hidden, used by scripts or shown only in particular interface states.

Within body there are particular elements called semantic elements (WHATWG, 2026c). Semantic elements describe the role of the different parts of the page, allowing software agents and assistive technologies to better understand its organisation (WHATWG, 2026c; W3C, 2026b).

These elements are:

Element	Function
<header>	Contains introductory information about the page or section in which it is located. It can include a heading, logo, subtitle, author, search or navigation.
<nav>	Identifies an important navigation block, such as the main menu, the page's table of contents or navigation between sections. It does not need to be used for every small group of links.
<main>	Contains the main, distinctive content of the page, normally excluding the header, menus, sidebars and footer. Generally, there should be only one visible <main>.
<article>	Represents self-contained content that could be understood or distributed separately, such as an article, a news item, a post, a review or a comment.
<section>	Groups content belonging to the same topic. It normally has a heading and represents part of a larger piece of content, such as a chapter or a thematic section.
<aside>	Contains information only indirectly related to the surrounding content, such as further information, advertising, quotations, related links or a sidebar.
<footer>	Contains concluding information about the page or section in which it is located, such as the author, publication date, copyright, contact details or secondary links.

DOM

DOM stands for Document Object Model (WHATWG, 2026a).
When the browser interprets an HTML document, it creates a structured representation of its contents (WHATWG, 2026a, 2026c).
This representation can be viewed as a tree of nodes (WHATWG, 2026a):

- Each *HTML* element, such as paragraphs, headings, images, links, etc., is represented by a node (WHATWG, 2026a, 2026c).
- Nodes are organised hierarchically so that nested elements are children of their parent elements (WHATWG, 2026a).

For example, the code:

```
<body>
  <h1>Heading</h1>
  <p>Paragraph text.</p>
</body>
```

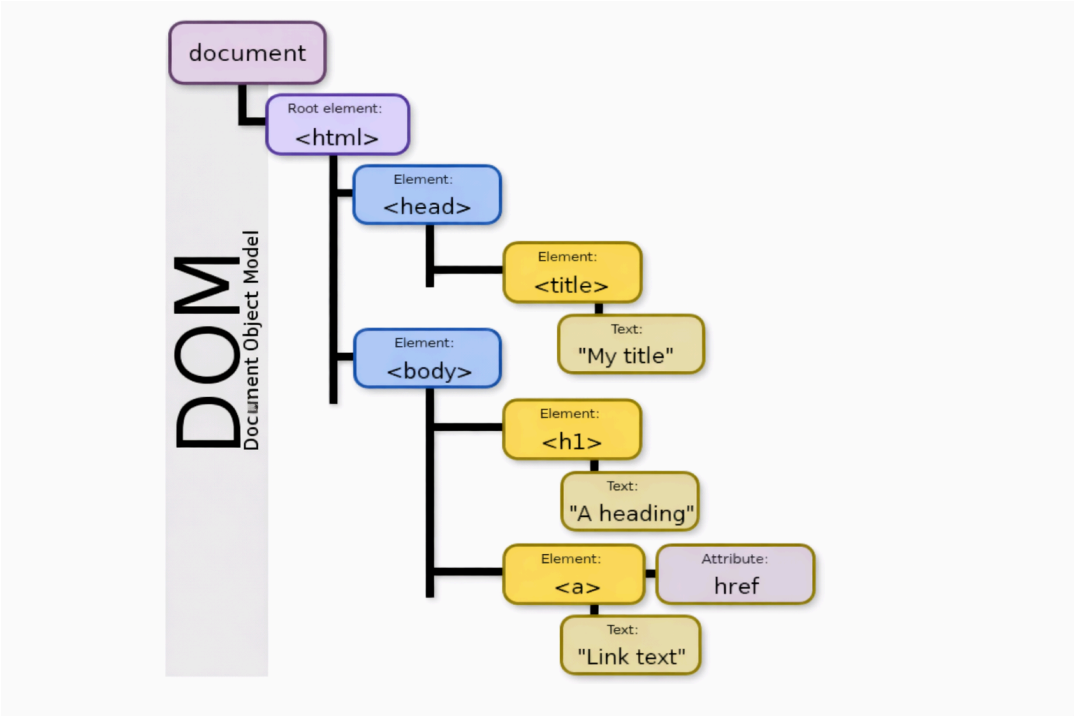
can be represented in simplified form as follows:

```
Document
└── html
    ├── head
    └── body
        ├── h1
        │   └── "Heading"
        └── p
            └── "Paragraph text."
```

In the *DOM* we find, among others:

- Document nodes.
- Element nodes.
- Text nodes.
- Comments.

Attributes are associated with elements and can be read or modified (WHATWG, 2026a, 2026c).
The DOM Standard defines the document as a hierarchically organised structure of nodes (WHATWG, 2026a).



The *DOM* allows the browser and JavaScript programs to (WHATWG, 2026a, 2026c):

- Know the current structure of the document.
- Locate elements.
- Modify their content and attributes.
- Add or remove elements.
- Handle events, such as clicking a button or entering data.
- Update the interface without reloading the entire page.

In particular, the *DOM* allows JavaScript to dynamically modify the document represented in the browser (WHATWG, 2026a, 2026c).
JavaScript does not directly modify the *HTML* code received, but acts on the *DOM* nodes present in the browser's memory (WHATWG, 2026a, 2026c).
The *DOM* therefore represents the current structure and content of the page and may differ from the *HTML* code initially received (WHATWG, 2026a, 2026c).
The latter remains unchanged, while JavaScript can modify the *DOM* to dynamically add, remove or update the page's content.

Example

```
document.querySelector("h1").textContent = "New heading";
```

This JavaScript statement locates the first <h1> element in the *DOM* and changes its text content. The change is displayed on the page, but does not modify the original HTML document stored on the server.

The *DOM* is built and maintained in the browser process's RAM while the page is open (Chromium Projects, n.d.; WHATWG, 2026a).
The process can be summarised as follows:

- HTML data → browser parsing → *DOM* construction → display

When JavaScript modifies the document, the *DOM* in memory is updated (WHATWG, 2026a, 2026c).
The original HTML file stored on the server is not modified.
When the user closes the tab or visits another page, the *DOM* is normally removed from memory (Chromium Projects, n.d.; WHATWG, 2026c).

HTML example

Below is a very simple example of an *HTML* page containing a heading, a paragraph of text and an image (WHATWG, 2026c):

```
<!DOCTYPE html>
<html>

<head>
 <title>Example HTML Page</title>
</head>

<body>
 <h1>Welcome to my Web page!</h1>
 <p>This is an example HTML page</p>
 <img src="immagine.jpg" alt="Example image">
</body>

</html>
```

In this example:

- *<!DOCTYPE html>* mainly serves to make the browser use standards-compliant mode (WHATWG, 2026c).
- *<html>* is the root element that encloses all the content of the page (WHATWG, 2026c).
- *<head>* contains metadata and document information/settings, as correctly explained on the preceding pages (WHATWG, 2026c).
- *<title>* defines the title of the document, and the browser can use it to identify the page in the tab, window, history or bookmarks (WHATWG, 2026c).
- *<body>* contains the main content of the page, such as text and images (WHATWG, 2026c).
- *<h1>* is a heading tag that defines a first-level heading (WHATWG, 2026c).
- *<p>* is a paragraph tag that contains text (WHATWG, 2026c).
- *<img>* is used to insert an image into the page. The src attribute specifies *the URL* of the resource to display. In the example, immagine.jpg is a relative *URL* with no other

path components, so it indicates a file located in the same directory as the HTML document. The alt attribute, on the other hand, provides alternative text describing the image, used when the image cannot be displayed and by assistive technologies (WHATWG, 2026c, 2026d; W3C, 2024b).



The page generated from our example html with an image of a beaver

Supplementary material

Exercises

Correcting an HTML page
A student writes this code:

```
<p>
  This is an <strong>important text
</p>
</strong>

<img src="montagna.jpg"></img>

<a>Visit the site</a>
```

Analyse the code.

- Identify all the errors present.
- Rewrite the paragraph with the element correctly.
- Explain why elements must be nested correctly.
- Correct the <img> element and explain why it does not have a closing tag.
- Add the attributes needed to indicate the resource and provide alternative text to the image.
- Correct the link so that it leads to https://example.com.
- For the corrected link, identify the opening tag, attribute, value, content and closing tag.
- Finally, explain the difference between a tag and an HTML element.

Building the structure of a page
You need to create a simple page dedicated to a photography exhibition.

The page must contain:

- The main heading “Photography exhibition”.
- A short introductory text.
- A section dedicated to the authors.
- A section dedicated to the works.
- A navigation menu.
- A sidebar with further information.
- Concluding information about the author, copyright and contact details.

Write a possible HTML structure using, where appropriate:
<header>, <nav>, <main>, <section>, <article>, <aside>, <footer>, <h1> and <p>.

Then:

- Explain why you chose each semantic element.

- Identify which element should contain the main content of the page.
- Explain why there should normally be only one visible <main>.
- Indicate what content you would place in <aside>.
- Explain the difference between <section> and <article> using the exhibition example.
- Finally, represent the page structure as a hierarchical tree of elements.
- Explain why a correct semantic structure is useful not only for the browser, but also for search engines and assistive technologies.

Completing an HTML document
You are given this incomplete document:

```
<html>
<head>
    <title>School website</title>
</head>

<body>
    <h1>Welcome</h1>
    <p>This is our school’s website.</p>
</body>
</html>
```

Complete and improve the document.
- Add the appropriate DOCTYPE.
- Indicate in the <html> element that the main language is Italian.
- Add the UTF-8 encoding declaration.
- Add the viewport configuration suitable for mobile devices.
- Explain the function of each of the four additions.
- Explain the difference between <head> and <body>.
- Explain why <title> does not coincide with <h1>.
- Indicate where the browser might use the content of <title>.
- Explain what problem could occur on smartphones without the correct viewport declaration.

Block or inline?
Consider this fragment:

```
<p>
  Visit our
  <a href="https://example.com">official website</a>
  to read all the information.
</p>

<div>First area</div>
<div>Second area</div>
```

```
<p>
  This text contains a word
  <strong>important</strong>.
</p>
```

Answer the following questions:

- Classify <p>, <a>, and as block or inline elements according to the default behaviour described in the handout.
- Explain how the two elements will normally be arranged.
- Explain why the link can appear on the same line as the text surrounding it.
- Explain why does not normally cause a new line to begin.
- Indicate the fundamental difference between the default behaviour of a block element and that of an inline element.
- A student claims: “A element will always be block”. Evaluate the statement.
- Explain which technology allows the default visual behaviour of HTML elements to be changed.

HTML and DOM are not the same thing

The server sends this document to the browser:

```
<body>
    <h1>Available products</h1>
    <p>3 products found.</p>
</body>
```

After loading, a JavaScript program performs operations that:

- Change the heading to “Products on sale”.
- Change the paragraph text to “5 products found”.
- Add a new button to the page.

Answer the following questions:

- Represent the initial DOM as a simple tree.
- Explain how the DOM changes after JavaScript execution.
- Explain why the DOM is a hierarchical representation made up of nodes.
- Indicate which types of node appear in the example.
- Explain why JavaScript can change what the user sees without directly modifying the original HTML file on the server.
- Then explain why the current DOM may differ from the HTML initially received.
- Indicate where the DOM is normally maintained while the page is open.
- Explain what normally happens to the DOM when the user closes the tab.
- Use the example to reconstruct the sequence HTML received → browser parsing → DOM → modification through JavaScript → interface update.

Bibliography

Chromium Projects. (n.d.). *Multi-process architecture*. https://www.chromium.org/developers/design-documents/multi-process-architecture/

WHATWG. (2026a). *DOM Standard*. https://dom.spec.whatwg.org/

WHATWG. (2026b). *Encoding Standard*. https://encoding.spec.whatwg.org/

WHATWG. (2026c). *HTML Living Standard*. https://html.spec.whatwg.org/

WHATWG. (2026d). *URL Standard*. https://url.spec.whatwg.org/

World Wide Web Consortium. (2024a). *CSS Viewport Module Level 1*. https://www.w3.org/TR/css-viewport-1/

World Wide Web Consortium. (2024b). *Web Content Accessibility Guidelines (WCAG) 2.2*. https://www.w3.org/TR/WCAG22/

World Wide Web Consortium. (2026a). *CSS Display Module Level 3*. https://www.w3.org/TR/css-display-3/

World Wide Web Consortium. (2026b). *HTML Accessibility API Mappings 1.0*. https://www.w3.org/TR/html-aam-1.0/

Further reading

- Jennifer Niederst Robbins, Learning Web Design: A Beginner's Guide to HTML, CSS, JavaScript, and Web Graphics, 6th ed., O'Reilly Media, 2025 — Probably the best overall reference for this handout. It provides a systematic introduction to HTML syntax, document structure, elements, attributes, links, images, semantic markup, and accessibility. Its progressive approach and practical exercises make it particularly suitable for students who are learning how to construct HTML documents for the first time.
- Jon Duckett, HTML & CSS: Design and Build Websites, Wiley, 2011 — A highly visual introduction to HTML and CSS, explaining how web pages are structured through elements, tags, attributes, headings, paragraphs, links, images, and hierarchical relationships. Its diagrams and practical examples make it especially useful for understanding the fundamental distinction between HTML structure and CSS presentation.
- Matthew MacDonald, HTML5: The Missing Manual, 2nd ed., O'Reilly Media, 2013 — Explores the capabilities of modern HTML, with particular attention to semantic elements, document organization, forms, multimedia, and web applications. It is especially relevant to the handout's explanation of elements such as <header>, <nav>, <main>, <article>, <section>, <aside>, and <footer>, showing how semantic markup can communicate the logical organization of a document. Some implementation details are dated and should be checked against current HTML standards.
- Laura Kalbag, Accessibility for Everyone, A Book Apart, 2017 — Provides a practical introduction to accessible web design, emphasizing meaningful HTML, properly structured documents, readable content, and compatibility with assistive technologies. It is particularly useful for understanding why semantic markup, headings, language declarations, image alternatives, and logical document structure matter beyond visual presentation.

- Cody Lindley, DOM Enlightenment, O'Reilly Media, 2013 — The most specialized reference for the handout's section on the Document Object Model. It explains how HTML documents are represented as hierarchical structures of nodes and how JavaScript can access, traverse, create, modify, and remove these nodes. It is particularly relevant to the distinction between the original HTML document and the DOM maintained in browser memory, which can change dynamically during interaction.

Editorial note

To produce these materials, I used generative artificial intelligence tools to support writing, particularly to improve the form of the text, reorganise the content, correct wording and speed up certain editorial tasks.

Working independently on the production of these materials, I aim to automate all activities that can be automated, so that I can devote more time to research, design and content development.

Artificial intelligence does not, however, determine the content of the work: the choice of topics, the structure, the ideas, the interpretations, the examples and the teaching approach are developed by me. AI is therefore used mainly as a tool to support production and formal revision, while authorship and responsibility for the design of the content remain mine.

Unless otherwise stated, this material is distributed under the Creative Commons Attribution – NonCommercial – ShareAlike 4.0 International licence (CC BY-NC-SA 4.0).

It is therefore possible to share, redistribute, modify and create derivative works from the material, provided that it is properly attributed, that it is not used for commercial purposes and that any modified or derivative versions are distributed under the same licence.

