Text basics

- Headings, revisited
- Quotes and citations
  - HTML Entities

Similar to how your text editor provides `<h1>` to `<h6>` headings, along with a plethora of ways to format sections of text in meaningful and visual ways, HTML provides a very similar set of semantic and non-semantic elements to make meaning of prose.

This section covers the main ways of marking up text, or text basics. We will then discuss attributes, before exploring additional ways of marking up text, such as lists, tables, and forms.

Headings, revisited

There are six section heading elements, `<h1>`, `<h2>`, `<h3>`, `<h4>`, `<h5>`, and `<h6>`, with `<h1>` being most important and `<h6>` the least. For many years, developers were told that headings were used by browsers to outline documents. That was originally a goal, but browsers haven't implemented outlining features. However, screen reader users do use headings as an exploration strategy to learn about the content of the page, navigating through headings with the `h` key. So ensuring that heading levels are implemented as you would outline a document makes your content accessible and is still very much encouraged.

By default, browsers style `<h1>` the largest, `<h2>` slightly smaller, with each subsequent heading level being smaller by default. Interestingly, browsers by default also decrement the `<h1>` font size based on how many `<article>`, `<aside>`, `<nav>`, or `<section>` elements it is nested in.

Nested H1 examples.

Some user agent stylesheets include the following selectors, or similar, to style nested `<h1>` elements as if they were of a less important level:

```
h2, :is(article, aside, nav, section) h1 {}
h3, :is(article, aside, nav, section) :is(article, aside, nav, section) h1 {}
```

But the Accessibility Object Model, or AOM, still reports the level of the element correctly; in this case, "heading, level 1". Note that the browser doesn't do this for other heading levels. That said, don't use heading level-based browser styling. Even though browsers don't support outlining, pretend they do; mark up your content headings as if they do. That will make your content make sense to search engines, screen readers, and future maintainers (which just might well be you).

Outside of headings, most structured text is made up of a series of paragraphs. In HTML, paragraphs are marked up with the `<p>` tag; the closing tag is optional but always advised.

The #about section has a heading and a few paragraphs:

https://codepen.io/web-dot-dev/pen/jOKdjQm

This section is not a landmark as it doesn't have an accessible name. To turn this into a `region`, which is a landmark role, you can use `aria-labelledby` to provide the accessible name:

```
<section id="about"  aria-labelledby="about_heading">
<h2 id="about_heading">What you'll learn</h2>
```

Only create landmarks if and when appropriate. Having too many landmarks can quickly become disorienting for screen reader users.

Quotes and citations

When marking up an article or blog post, you may want to include a quote or pull-quote, with or without a visible citation. There are elements for these three components: `<blockquote>`, `<q>`, and `<cite>` for a visible citation, or the `cite` attribute to provide more information for search.

The `#feedback` section contains a header and three reviews; these reviews are blockquotes, some of which contain quotes, followed by a paragraph containing the quote's citation. Omitting the third review to save space, the markup is:

https://codepen.io/web-dot-dev/pen/eYKxwbV

The information about the quote author, or citation, is not part of the quote and therefore not in the `<blockquote>`, but comes after the quote. While these are citations in the lay sense of the term, they are not actually citing a specific resource, so are encapsulated in a `<p>` paragraph element.

The citation appears over three lines, including the author's name, previous role, and professional aspiration. The `<br>` line break creates a line break in a block of text. It can be used in physical addresses, in poetry, and in signature blocks. Line breaks should not be used as a carriage return to separate paragraphs. Instead, close the prior paragraph and open a new one. Using paragraphs for paragraphs is not only good for accessibility but enables styling. The `<br>` element is just a line break; it is impacted by very few CSS properties.

While we provided citation information in a paragraph following each blockquote, the quotes shown earlier are coded this way because they didn't come from an external source. If they did, the source can (should?) be cited.

If the review was pulled from a review website, book, or other work, the `<cite>` element could be used for the title of a source. The content of the `<cite>` can be the title of a book, the name of a website or TV show, or even the name of a computer program. The `<cite>` encapsulation can be used whether the source is being mentioned in passing or if the source is being quoted or referenced. The content of the `<cite>` is the work, not the author.

If the quote from Blendan Smooth was taken from her offline magazine, you would write the blockquote like this:

https://codepen.io/web-dot-dev/pen/LYrqKMq

The citation element `<cite>` has no implicit role and should get its accessible name from its contents; don't include an `aria-label`.

To provide credit where credit is due when you can't make the content visible, there is the `cite` attribute which takes as its value the URL of the source document or message for the information quoted. This attribute is valid on both `<q>` and `<blockquote>`. While it's a URL, it is machine readable but not visible to the reader:

https://codepen.io/web-dot-dev/pen/NWzoZob

While the `</p>` closing tag is optional (and always recommended), the `</blockquote>` closing tag is always required.

Most browsers add padding to both `<blockquote>` inline directions and italicize `<cite>` content; this can be controlled with CSS. The `<blockquote>` does not add quotation marks, but those can be added with CSS-generated content. The `<q>` element does add quotes by default, using language-appropriate quotation marks.

In the `#teachers` section, HAL is quoted as saying, "I'm sorry <NAME REDACTED, RIP>, but I'm afraid I can't do that, ." The code for that, in English and French, is:

https://codepen.io/web-dot-dev/pen/qBKgzgg

The inline quotation element, `<q>`, adds language-appropriate quotes. The user-agent default styles include open-quote and close-quote generated content:

```
q::before {content: open-quote;}
q::after {content: close-quote;}
```

The `lang` attribute is included to let the browser know that, while base language of the page was defined as English in the `<html lang="en-US">` opening tag, this paragraph of text is in a different

language. This helps voice controls such as Siri, Alexa, and voiceOver use French pronunciation. It also informs the browser what type of quotes to render.

Like `<blockquote>`, the `<q>` element supports the `cite` attribute.

https://codepen.io/web-dot-dev/pen/jOKdjJw

HTML Entities

You may have noticed the escape sequence or "entity". Because the `<` is used in HTML, you have to escape it using either `<` or a less easy-to-remember encoding `<`. There are four reserved entities in HTML: `<`, `>`, `&`, and `"`. Their character references are `<`, `>`, `&` and `"` respectively.

A few other entities you will often use are `©` for copyright (©), `™` for Trademark (™), and ` ` for non-breaking space. Non-breaking spaces are useful when you want to include a space between two characters or words while preventing a line break from occurring there. There are over 2,000 named character references. But, if needed, every single character, including emojis, has an encoded equivalent that starts with `&#`.

If you take a look at ToastyMcToastface's workshop review (not included in the code sample above), there are some unusual text characters:

```
<blockquote>Learning with Hal and Eve exceeded all of my wildest fantasies. All they did was stick a USB in. They promised that it was a brand new USB, so we know there were no viruses on it. The Russians had nothing to do with it. This has no—□t̨h̨□̨įn□̨□̨g□̨ □̨□̨t̨o□̨ □̨o̧□̨□̨dó□̨□̨ □̨□□̨̨́w□̨□□̨ìth□̨ □̨□̨</blockquote>
```

The last sentence in this blockquote can also be written as:

```
This has
no&#x336;&#x33C;&#x356;&tcedil;&#x318;h&#x31D;&#x330;&#x329;&#x348;&#x317;i&#x319;&#x32A;n&#x34F;&#x329;&#x319;
&#x34D;&#x331;&#x32B;&#x31C;&#x31F;g&#x322;&#x323;&#x345;
&#x317;&#x330;&#x353;&#x332;&#x31E;&#x300;t&#x359;&#x300;o&#x31F;
&#x316;&#x356;&#x339;&#x315;
&#x353;&#x33C;&#x34E;&#x31D;&#x356;&#x32D;d&oacute;&#x32A;&#x320;&#x355;&#x31C; &#x34D;&#x331;
&#x34E;&#x35A;&#x32F;&#x31F;&#x301;w&#x32E;&#x332;&#x339;&#x355;&#x348;&#x31F;&#x35E;&igrave;t
h&#x322; &#x330;&#x333;
&#x32F;&#x32E;&#x347;
```

There are a few unescaped characters and a few named character references in this code mess. Because the character set is UTF-8, the last few characters in the blockquote don't actually need to be escaped, as in this example. Only characters not supported by the character set need to be escaped. If needed, there are many tools to enable escaping various characters, or you can just ensure you include `<meta charset="UTF-8">` in the `<head>`.

Even when you specify the character set as UTF-8, you still have to escape the `<` when you want to print that character to the screen. Generally, you don't need to include the named character references for `>`, `"`, or `&`; but if you want to write a tutorial on HTML entities, you do need to write `<` when teaching someone how to code a `<`. □□

Oh, and that smiley emoji is `😀`, but this doc is declared as UTF-8, so it isn't escaped.

□□□ #3
□ □□□ □□□ 2026-10-31 22:51:59 UTC
□ cocobook-sync □□□ 2026-11-04 17:39:25 UTC