Anatomy of an HTML Tag

clearlydecoded.com/anatomy-of-html-tag

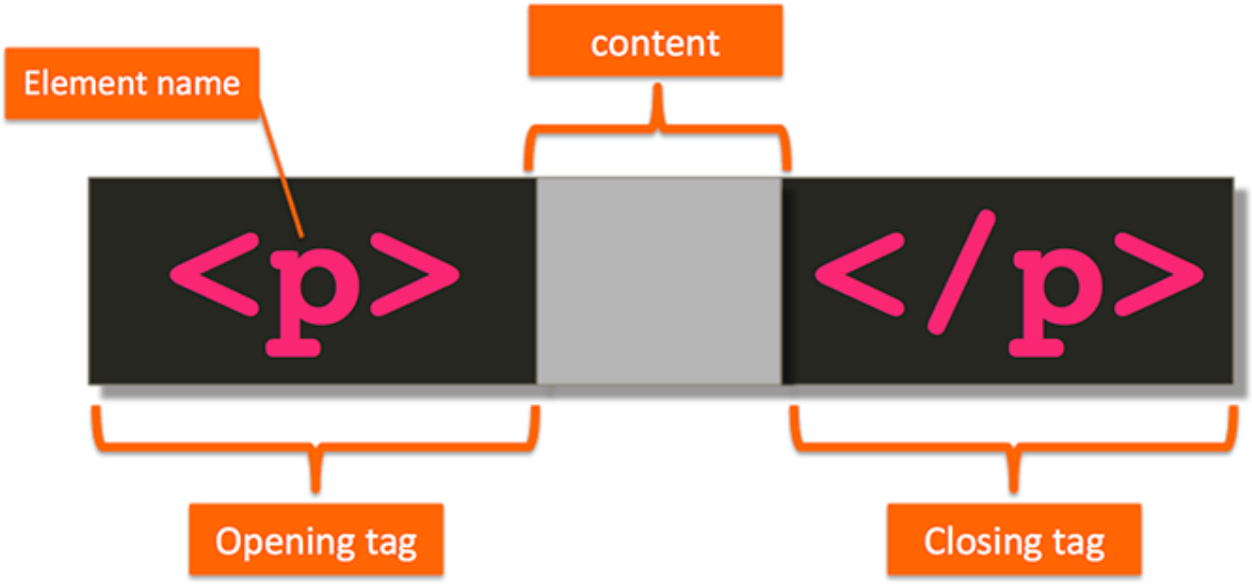
Yaakov Chaikin

As mentioned in one of the previous articles, HTML is a language. Each language comes with its own grammar or, as it's referred to in the context of programming, its own *syntax*.

Come On and *Tag* Along! 😁

Since HTML is a tag-based language, it's no surprise that at the core of HTML is the *HTML tag*. Let's look at the syntax of an HTML tag.

Usually, an HTML tag consists of an opening and a closing tag that surrounds some content, marking up or annotating that content as shown below.

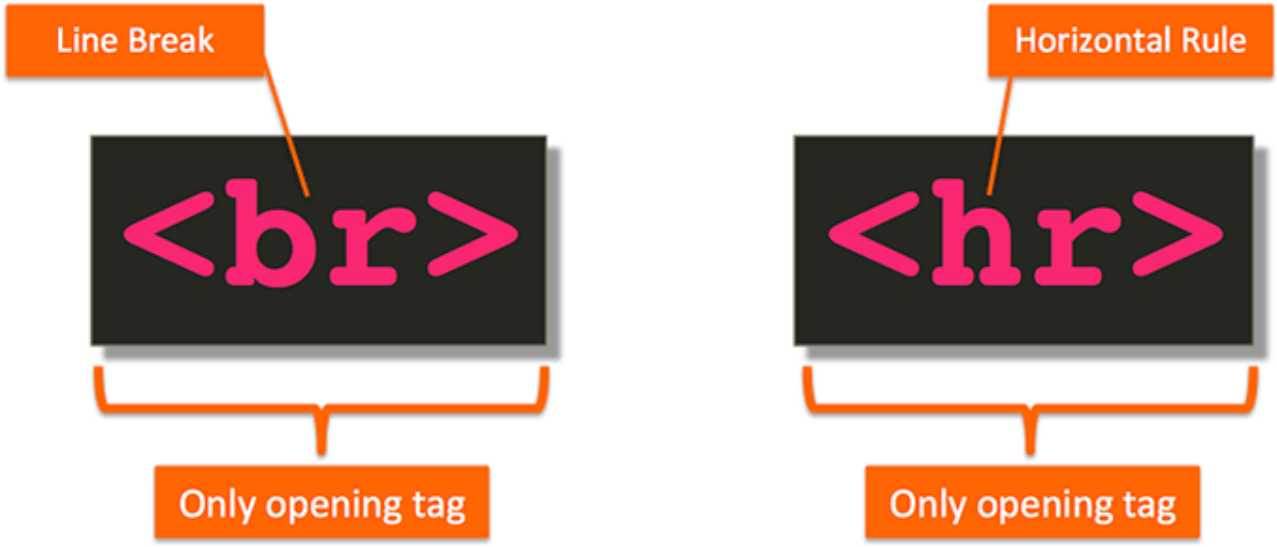


In this case, the tag `p`, which stands for paragraph, is communicating to us that the content in the gray area should be treated as a paragraph.

> Technically speaking, `p` by itself is called an *element*, and, together with the angle brackets, it's called a *tag*. However, while technically incorrect, people aren't usually careful to make that distinction and use these terms interchangeably.

Permitted Content

Most HTML tags have a closing tag. As you just saw, the opening tag `<p>` has a closing tag `</p>`. However, not all of them do. For example, as shown below, the `<br>` and `<hr>` tags only have an opening tag (`br` stands for line break, and `hr` stands for horizontal rule). They don't have a closing tag at all.



In fact, the following code snippet would be invalid HTML:

```
...
<br>Some
content</br>
...
```

The reason? The `<br>` tag tells the browser to create a line break in text, i.e., anything that comes *after* the `<br>` tag should be displayed on the next line. It would *never* make sense for the `<br>` tag to surround some content and therefore it's not allowed to.

The story is similar with the `<hr>` tag. The linked MDN page describes its *Permitted content* as *None, it is an empty element*.

The Self-Closing Tag

If you have come from the world of XML or ever used XHTML (previous version of HTML), you may have heard of the concept of a *self-closing tag*.

In XML and XHTML, a self-closing tag is a shorthand notation for an opening and closing tag in one. It's used to communicate lack of content in between the opening and closing tags. So, rather than typing `<p></p>` (with no space at all in between), you'd be able write `<p/>`.

In modern HTML, using self-closing tags are not allowed. The reason is that, semantically, there is a difference between an *inherently empty element* and an element that is permitted to have content but doesn't have any at the moment. The self-closing `<p/>` construct doesn't clearly communicate that as much as `<p></p>`.

I know. Super philosophically technical. 😉

However, you do need to know that if you use a self-closing tag like `<p/>` in your HTML, the browser will ignore the `/` and interpret it as a starting tag `<p>`.

Close What You Open

If you came from the strict world of XHTML, it might come as a surprise to you that in a lot circumstances, modern HTML allows you to omit the closing tag altogether.

However, the rules of when it's ok to omit the closing tag and how to properly communicate to the browser what content the opening tag is supposed to apply to are rather tedious. Most importantly, the HTML code becomes fairly hard to read.

When does one paragraph end and the other one begin? Not 100% clear unless you happen to remember those rules.

The solution is to simply comply with the following *best practice*:

> If a tag has a closing tag, ALWAYS use it.

Close First What You Opened Last

Ever double bag something, tying every bag in the process?

Think about how you would go about doing that.

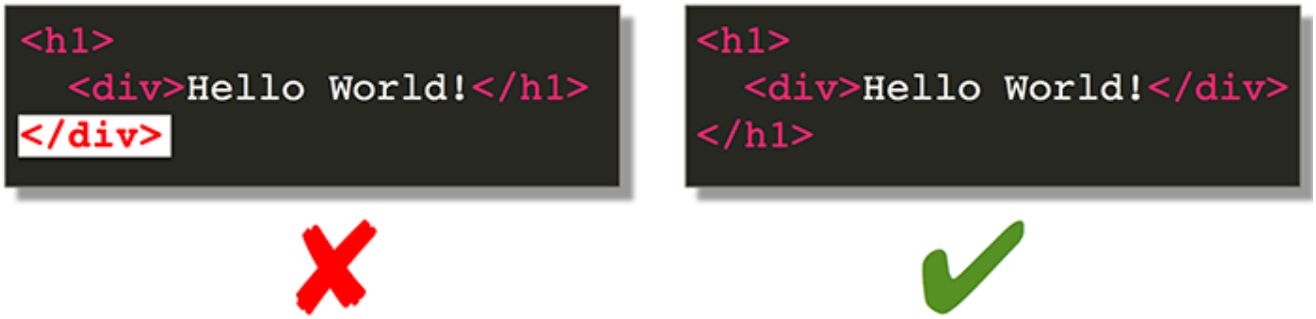
1. Place the item into *bag 1*
2. Tie *bag 1*
3. Place *bag 1* into *bag 2*
4. Tie *bag 2*

If, after placing *bag 1* into *bag 2*, you decided to *first* tie *bag 2*, you'd never be able to get to tie *bag 1* again!

Same goes for tag closing.

> The last tag opened is the first tag closed.

You've seen the two code snippets that communicate this idea when we discussed What is HTML? Here it is again:

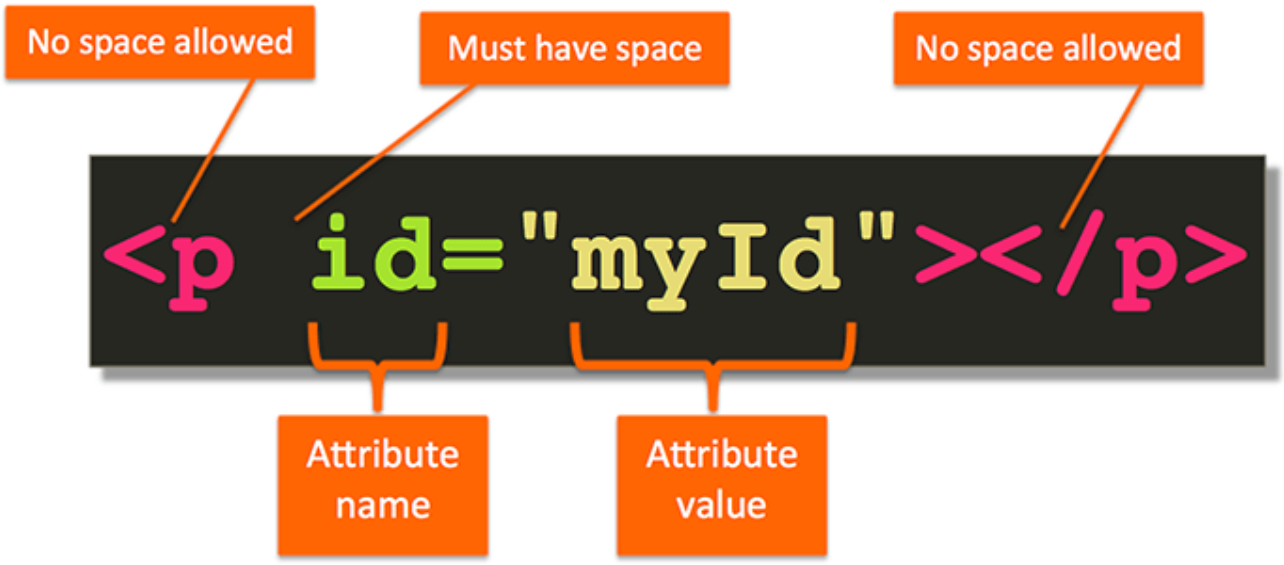


Since the `div` tag was *opened last*, it must be *closed first*, before the outer `h1` tag is closed.

Element Attributes

Every HTML element can have predefined attributes. An HTML attribute is a way to provide additional information about the tag its attached to. Think of attributes as a way to further customize the meaning and/or behavior of the tag.

Attributes go into the opening tag as name-value pair as is shown below, pointing out the spacing rules:



In the above example, the attribute's *name* is `id` and it's assigned `myId` as its *value*.

Each attribute has its own rules for the meaning of its value. For example, the value of the `id` attribute has to be unique in the entire HTML document. In other words, no other tag is allowed to have an `id` attribute with the same value.

Space 🚀🌎

Um... Not that kind of space. 😊

The above illustration does a pretty good job of showing where you must have spaces and where spaces are not allowed.

However, in addition to that, you should know the following:

> Wherever at least one space is allowed, placing extra spaces, including new line characters, tabs, etc. are allowed as well. These extra spaces do not make any difference as they are ignored by the browser.

For example, this is perfectly legal HTML:

```
...
<p id="myId"
  style="..."
  class="..."
              something-else="..."




              even-this="..."
      >
Hey! I got dizzy trying to read this
code!!
</p>
...
```

Obviously, just because it’s allowed, doesn’t mean your code formatting should be as crazy as what I did there. With that type of formatting, you are risking being “admired” by your fellow developers who have to read it later.

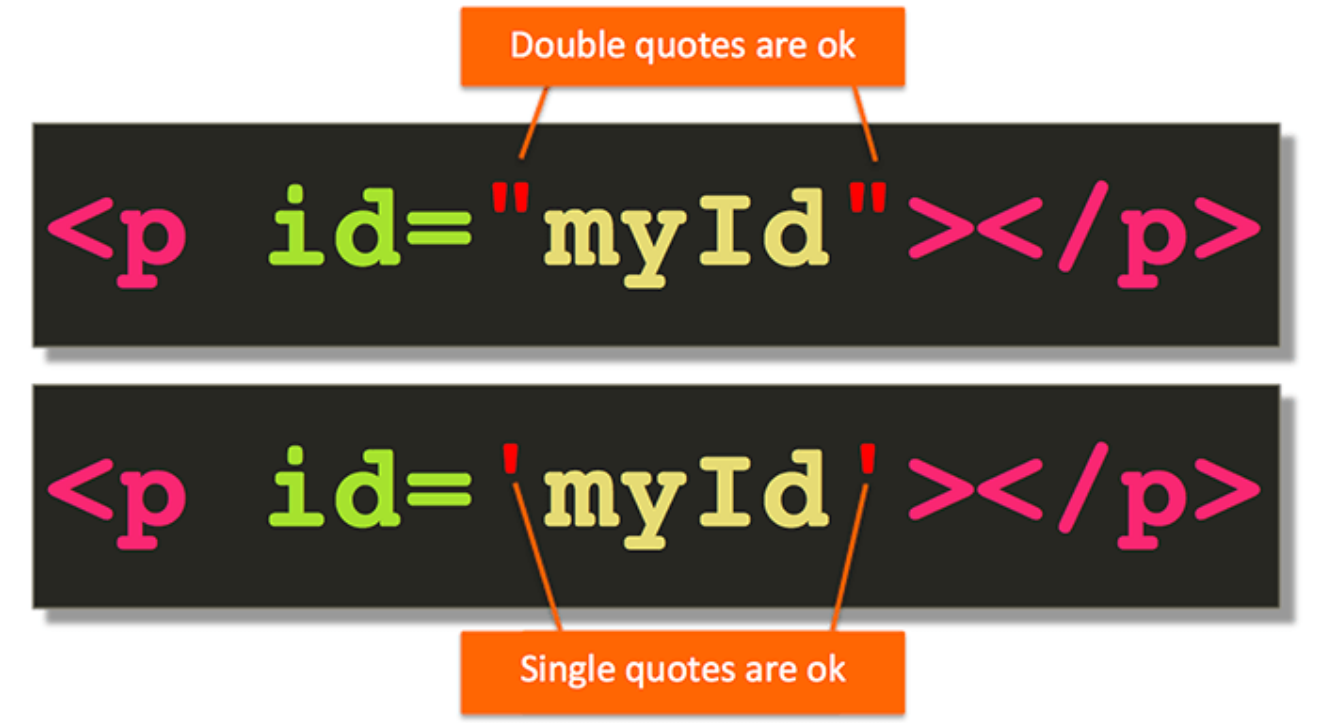
Instead, use the extra spaces to make it *easier* to read the code:

```
...
<p id="myId"
  style="..."
  class="..."
  something-else="..."
  even-this="...">
Who wrote this code? It's SOOOOO
beautiful! 😍
</p>
...
```

Quotable Quotes

Enclosing the value of an attribute in quotes is technically not required in all circumstances. Nevertheless, it’s best practice to always surround the value of the attribute in either single or double quotes.

It doesn’t matter whether you use single or double quotes. They are equivalent in HTML.

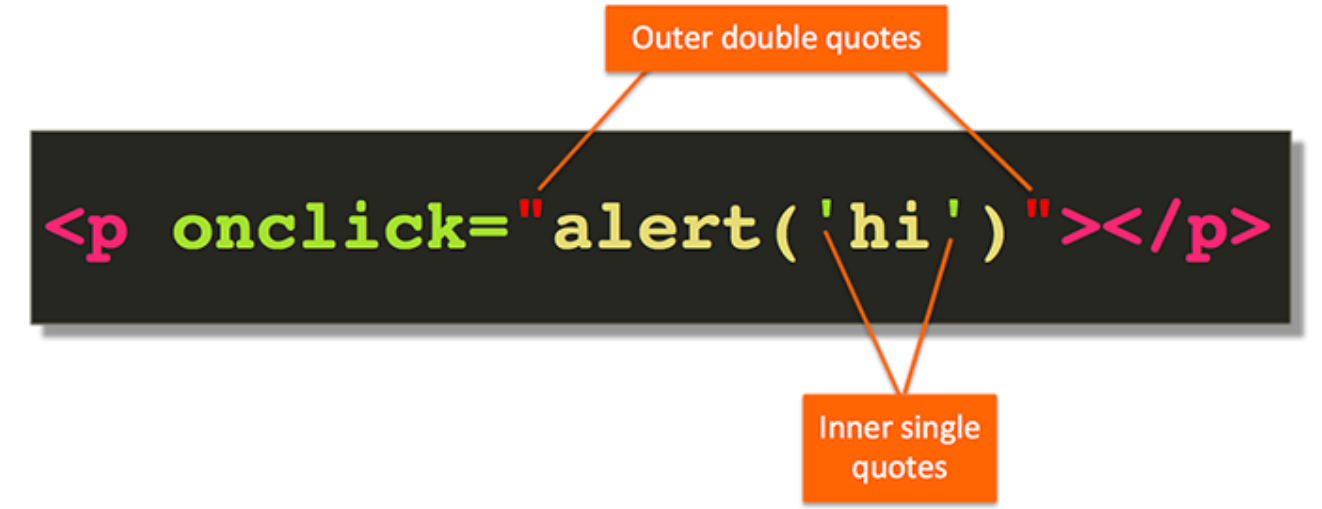


A more interesting case arises when the attribute value itself contains quotes. In other words, the value content actually contains either single or double quotes. For example, the following code would be problematic:

```
<p onclick="alert("hi")"></p>
```

If you leave the code as is shown above, the browser will see the value of the attribute `onclick` as `alert(` . That's because after the parser sees the opening `"` , it looks for the matching `"` to terminate the opening quote. The browser will then likely get confused as to what to do with the rest of the value and interpret it as a bad attempt at another attribute insertion. Not good.

The solution is to interchange double and single quotes such that you close the quotes in the opposite order of opening them. So, if the last quote was a single quote, it must be closed first. Which quotes you start with doesn't make any difference:



You can *not* nest these as many times as you want. Two levels is as far as you can take it (as shown). If you required yet another set of inner quotes, you'd use the HTML character entity reference (e.g., `&quot;`), but that's for another article. In practice, it's rare to find more than two levels of nested quotes that you, as the HTML author, would want to manually specify.

Summary

Let's give a quick summary of what we've covered in this article:

- Most HTML tags have an opening and closing tag, but not all
- Tags that don't have a closing tag are not allowed to wrap content
- Self-closing tags (e.g., `<p/>`) are not allowed in modern HTML
- If a tag has a closing tag, always use it, even when it's technically optional
- When nesting tags, you must close the inner opened tag before you attempt to close the outer tag
- Attributes are name-value pairs that provide some additional information about the tag
- Where spaces are allowed, extra spaces are ignored by browsers, including tabs, new line characters, etc.
- When the attribute value itself contains quotes, interchange outer and inner quotes such that the last type of opened quote is terminated first