HTML basics

Previous | Overview: Getting started with the web | Next

HTML (Hypertext Markup Language) is the code that is used to structure and display a web page and its content. For example, content could be structured within a set of paragraphs, a list of bulleted points, or using images and data tables. As the title suggests, this article will give you a basic understanding of HTML and what its function is.

So what is HTML, really?

HTML is not a programming language; it is a *markup language*, and is used to tell your browser how to display the webpages you visit. It can be as complicated or as simple as the web designer wishes it to be. HTML consists of a series of **elements**, which you use to enclose, or wrap, different parts of the content to make it appear a certain way, or act a certain way. The enclosing tags can make a word or an image a hyperlink to somewhere else, can italicize words, and can make font bigger or smaller, and so on. For example, take the following line of content:

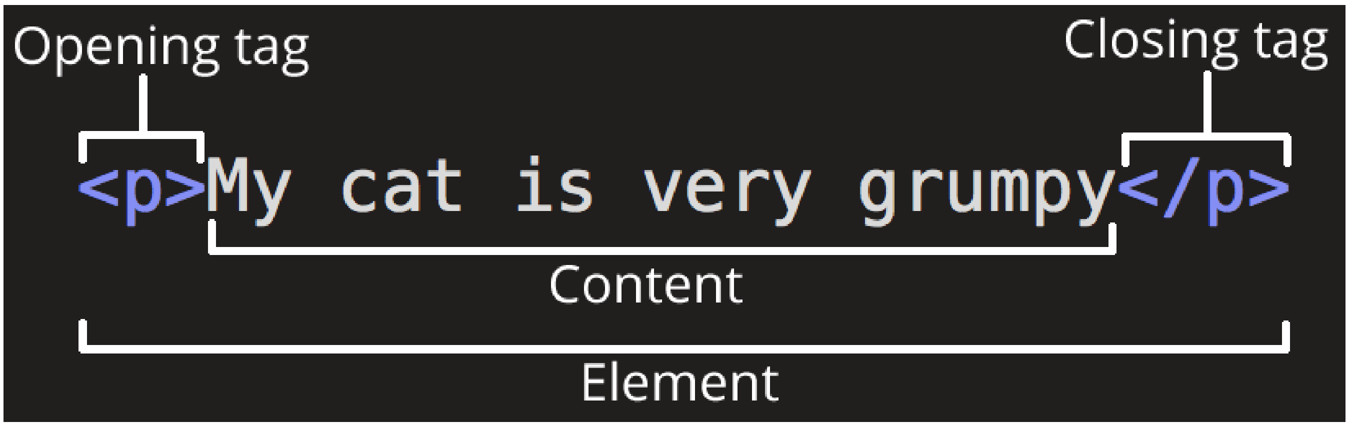
```
My cat is very grumpy
```

If we wanted the line to stand by itself, we could specify that it is a paragraph by enclosing it in a paragraph tag (`<p>`) element:

```
<p>My cat is very grumpy</p>
```

Anatomy of an HTML element

Let's explore this paragraph element a bit further.

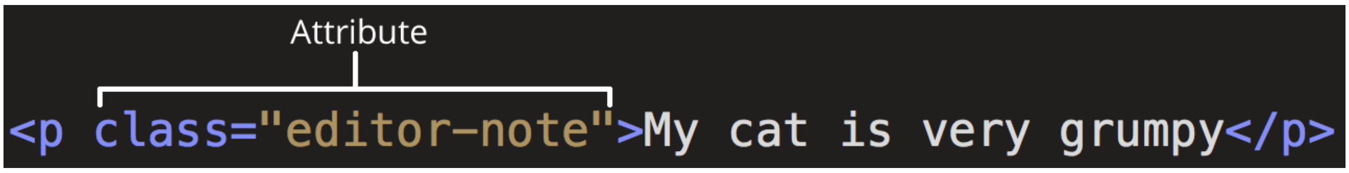


```
<p>My cat is very grumpy</p>
```

The main parts of our element are:

1. **The opening tag:** This consists of the name of the element (in this case, p), wrapped in opening and closing **angle brackets**. This states where the element begins, or starts to take effect — in this case where the start of the paragraph is.
2. **The closing tag:** This is the same as the opening tag, except that it includes a forward slash before the element name. This states where the element ends — in this case where the end of the paragraph is. Failing to include a closing tag is one of the common beginner errors and can lead to strange results.
3. **The content:** This is the content of the element, which in this case is just text.
4. **The element:** The opening tag, plus the closing tag, plus the content, equals the element.

Elements can also have attributes, which look like this:



```
<p class="editor-note">My cat is very grumpy</p>
```

Attributes contain extra information about the element which you don't want to appear in the actual content. In this case, the class attribute allows you to give the element an identifying name that can be later used to target the element with style information and other things.

An attribute should always have:

1. A space between it and the element name (or the previous attribute, if the element already has one or more attributes.)
2. The attribute name, followed by an equals sign
3. Opening and closing quote marks wrapped around the attribute value

Nesting elements

You can put elements inside other elements too — this is called **nesting**. If we wanted to state that our cat is VERY grumpy, we could wrap the word "very" in a `<strong>` element, which means that the word is to be strongly emphasized:

```
<p>My cat is <strong>very</strong> grumpy.</p>
```

You do however need to make sure that your elements are properly nested: in the example above we opened the <p> element first, then the <strong> element, therefore we have to close the <strong> element first, then the <p>. The following is incorrect:

```
<p>My cat is <strong>very grumpy.</p></strong>
```

The elements have to open and close correctly so they are clearly inside or outside one another. If they overlap like above, then your web browser will try to make a best guess at what you were trying to say, and you may well get unexpected results. So don't do it!

Empty elements

Some elements have no content, and are called **empty elements**. Take the `<img>` element we already have in our HTML:

```
<img src="images/firefox-icon.png" alt="My test image">
```

This contains two attributes, but there is no closing `</img>` tag, and no inner content. This is because an image element doesn't wrap content to have an effect on it. Its purpose is to embed an image in the HTML page, in the place it appears.

Anatomy of an HTML document

That wraps up the basics of individual HTML elements, but they aren't very useful on their own. Now we'll look at how individual elements are combined to form an entire HTML page. Let's revisit the code we put into our `index.html` example (which we first met in the Dealing with files article):

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>My test page</title>
  </head>
  <body>
    <img src="images/firefox-icon.png" alt="My test image">
  </body>
</html>
```

Here we have:

- `<!DOCTYPE html>` — the doctype. In the mists of time, when HTML was young (about 1991/2), doctypes were meant to act as links to a set of rules that the HTML page had to follow to be considered good HTML, which could mean automatic error checking and other useful things. However, these days no-one really cares about them, and they are really just a historical artifact that needs to be included for everything to work right. For now, that's all you need to know.
- `<html></html>` — the `<html>` element. This element wraps all the content on the entire page, and is sometimes known as the root element.
- `<head></head>` — the `<head>` element. This element acts as a container for all the stuff you want to include on the HTML page that *isn't* the content you are showing to your page's viewers. This includes things like keywords and a page description that you want to appear in search results, CSS to style our content, character set declarations, and more.
- `<body></body>` — the `<body>` element. This contains *all* the content that you want to show to web users when they visit your page, whether that's text, images, videos, games, playable audio tracks, or whatever else.
- `<meta charset="utf-8">` — this element sets the character set your document should use to utf-8, which includes most characters from all known human languages. Essentially it can now handle any textual content you might put on it. There is no reason not to set this, and it can help avoid some problems later on.
- `<title></title>` — this sets the title of your page, which is the title that appears in the browser tab the page is loaded in, and is used to describe the page when you bookmark/favourite it.

Images

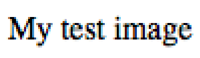
Let's turn our attention to the image element again:

```
<img src="images/firefox-icon.png" alt="My test image">
```

As we said before, it embeds an image into our page in the position it appears. It does this via the `src` (source) attribute, which contains the path to our image file.

We have also included an `alt` (alternative) attribute. In this attribute, you specify descriptive text for users who cannot see the image, possibly because:

1. They are visually impaired. Users with significant visual impairments often use tools called Screen Readers to read out the alt text to them.
2. Something has gone wrong causing the image to not display. For example, try deliberately changing the path inside your `src` attribute to make it incorrect. If you save and reload the page, you should see something like this in place of the image:


My test image

The key words about alt text are "descriptive text". The alt text you write should provide the reader with enough information to have a good idea of what the image conveys. In this example, our current text of "My test image" is no good at all. A much better alternative for our Firefox logo would be "The Firefox logo: a flaming fox surrounding the Earth."

Try coming up with some better alt text for your image now.

> **Note**: Find out more about accessibility at MDN's Accessibility landing page.

Marking up text

This section will cover some of the basic HTML elements you'll use for marking up text.

Headings

Heading elements allow you to specify that certain parts of your content are headings —or subheadings— of your content. In the same way that a book has a main title, chapter titles and subtitles, an HTML document can too. HTML contains six heading levels, `<h1>`–`<h6>` although you'll commonly only use 3–4 at most:

```
<h1>My main title</h1>
<h2>My top level heading</h2>
<h3>My subheading</h3>
<h4>My sub-subheading</h4>
```

Now try adding a suitable title to your HTML page just above your `<img>` element.

Paragraphs

As explained above, `<p>` elements are for containing paragraphs of text; you'll use these frequently when marking up regular text content:

```
<p>This is a single paragraph</p>
```

Add your sample text (you should have it from *What should your website look like?*) into one or a few paragraphs, placed directly below your <img> element.

Lists

A lot of the Web's content is lists and HTML has special elements for these. Marking up lists always consist of at least two elements. The most common list types are ordered and unordered lists:

1. **Unordered lists** are for lists where the order of the items doesn't matter, like a shopping list. These are wrapped in a `<ul>` element.
2. **Ordered lists** are for lists where the order of the items does matter, like a recipe. These are wrapped in an `<ol>` element.

Each item inside the lists is put inside an `<li>` (list item) element.

For example, if we wanted to turn the part of the following paragraph fragment into a list:

```
<p>At Mozilla, we’re a global community of technologists, thinkers, and builders
```

We could modfy the markup to this:

```
<p>At Mozilla, we’re a global community of</p>

<ul>
  <li>technologists</li>
  <li>thinkers</li>
  <li>builders</li>
</ul>

<p>working together ... </p>
```

Try adding an ordered or unordered list to your example page.

Links

Links are very important — they are what makes the Web A WEB. To add a link, we need to use a simple element — `<a>` — the *a* being short for "anchor". To make text within your paragraph into a link, follow these steps:

1. Choose some text. We chose the text "Mozilla Manifesto".
2. Wrap the text in an <a> element, like so:

   ```
   <a>Mozilla Manifesto</a>
   ```

3. Give the <a> element an href attribute, like so:

   ```
   <a href="">Mozilla Manifesto</a>
   ```

4. Fill in the value of this attribute with the web address that you want the link to link to:

   ```
   <a href="https://www.mozilla.org/en-US/about/manifesto/">Mozilla Manifesto
   ```

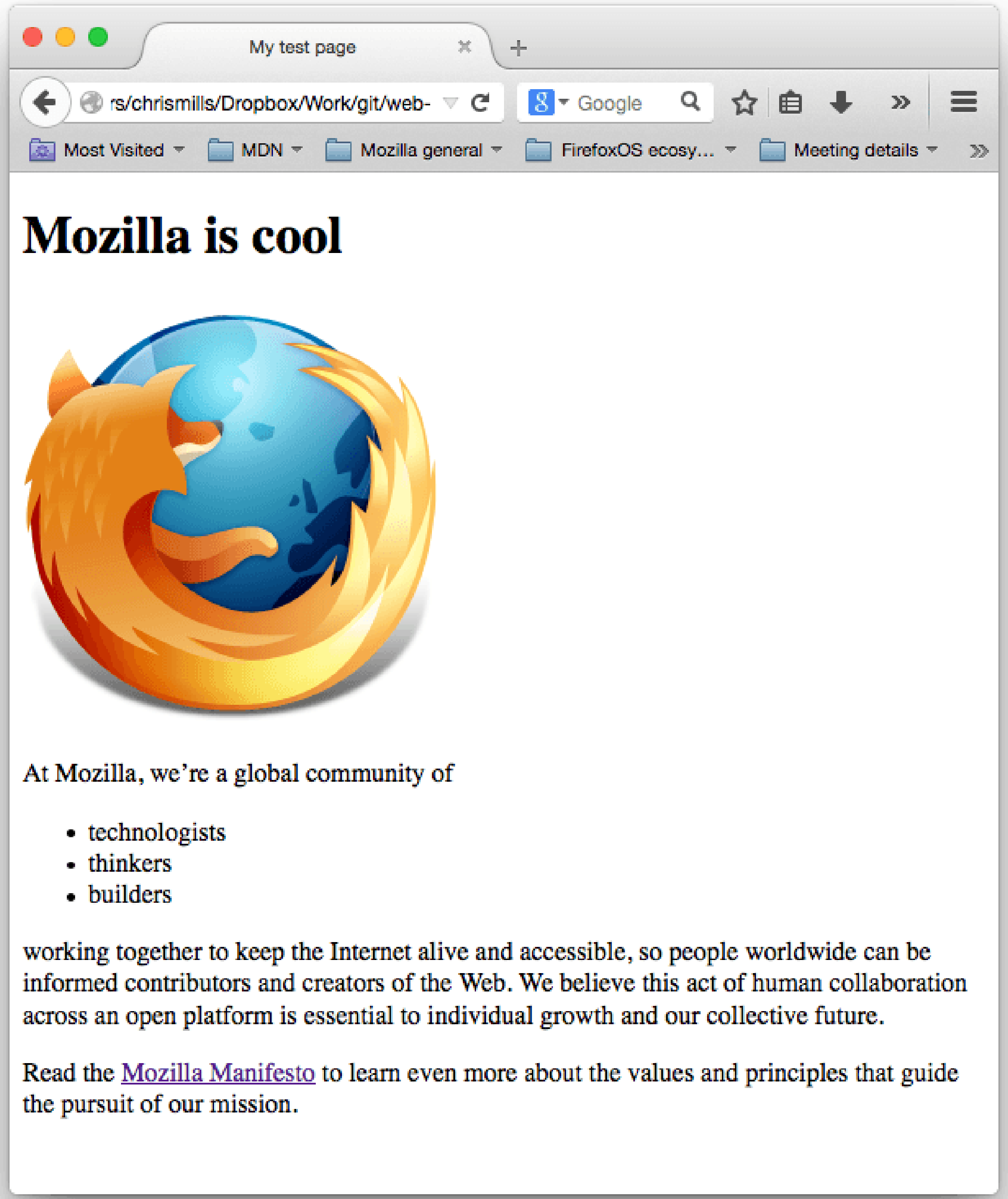
You might get unexpected results if you omit the `https://` or `http://` part, called the *protocol*, at the beginning of the web address. After making a link, click it to make sure it is sending you where you wanted it to.

`href` might appear like a rather obscure choice for an attribute name at first. If you are having trouble remembering it, remember that it stands for ***h**ypertext **ref**erence*.

Add a link to your page now, if you haven't already done so.

Conclusion

If you have followed all the instructions in this article, you should end up with a page that looks like the one below (you can also view it here):



Mozilla is cool

At Mozilla, we're a global community of

- technologists
- thinkers
- builders

working together to keep the Internet alive and accessible, so people worldwide can be informed contributors and creators of the Web. We believe this act of human collaboration across an open platform is essential to individual growth and our collective future.

Read the Mozilla Manifesto to learn even more about the values and principles that guide the pursuit of our mission.

If you get stuck, you can always compare your work with our finished example code on Github.

Here, we have only really scratched the surface of HTML. To find out more, go to our HTML Learning topic.

Previous Overview: Getting started with the web Next