One of HTML's main jobs is to give text structure and meaning (also known as semantics) so that a browser can display it correctly. This article explains the way HTML can be used to structure a page of text by adding headings and paragraphs, emphasizing words, creating lists, and more.

Prerequisites:	Basic HTML familiarity, as covered in Getting started with HTML.
Objective:	Learn how to mark up a basic page of text to give it structure and meaning—including paragraphs, headings, lists, emphasis, and quotations.

The basics: headings and paragraphs

Most structured text consists of headings and paragraphs, whether you are reading a story, a newspaper, a college textbook, a magazine, etc.

The Tribune Prints the News

Hilo Tribune.

We Lead — Others Follow

Vol. 10. HILO, HAWAII, HAWAIIAN ISLANDS, TUESDAY, FEBRUARY 28, 1905. No. 18.

The Hilo Tribune.

Drs. Grace and Irwin

Chas. M. LeBlond

Ridgway & Ridgway

Mortgagee's Notice of Intention to Foreclose.

KILAUEA CRATER CONTINUES IN ERUPTION

FOUR DAYS OF ACTIVITY

ANOTHER OUTBREAK OCCURS

LATER—Just as the TRIBUNE goes to press a telephone message from the Volcano House brings the news of renewed activity, more grand if possible than before. Parties returning from the crater report that the fire has burst through the landslide which choked the lava flow and a magnificent fountain is playing continuously, the extent of the opening in the side wall being 250 feet in length by 100 feet in width.

HONOLULU GETS APPROPRIATIONS FOR HARBOR AND LIGHTHOUSE

River and Harbor Bill Passes.

Postmaster Pratt Confirmed.

Latest Sugar Quotations.

Fire At New Orleans.

Structured content makes the reading experience easier and more enjoyable.

In HTML, each paragraph has to be wrapped in a `<p>` element, like so:

```
<p>I am a paragraph, oh yes I am.</p>
```

Each heading has to be wrapped in a heading element:

```
<h1>I am the title of the story.</h1>
```

There are six heading elements: `<h1>`, `<h2>`, `<h3>`, `<h4>`, `<h5>`, and `<h6>`. Each element represents a different level of content in the document; `<h1>` represents the main heading, `<h2>` represents subheadings, `<h3>` represents sub-subheadings, and so on.

Implementing structural hierarchy

For example, in this story, the `<h1>` element represents the title of the story, the `<h2>` elements represent the title of each chapter, and the `<h3>` elements represent sub-sections of each chapter:

```
<h1>The Crushing Bore</h1>

<p>By Chris Mills</p>

<h2>Chapter 1: The dark night</h2>

<p>It was a dark night. Somewhere, an owl hooted. The rain lashed down

<h2>Chapter 2: The eternal silence</h2>


<p>Our protagonist could not so much as a whisper out of the shadowy fi

<h3>The specter speaks</h3>

<p>Several more hours had passed, when all of a sudden the specter sat
```

It's really up to you what the elements involved represent, as long as the hierarchy makes sense. You just need to bear in mind a few best practices as you create such structures:

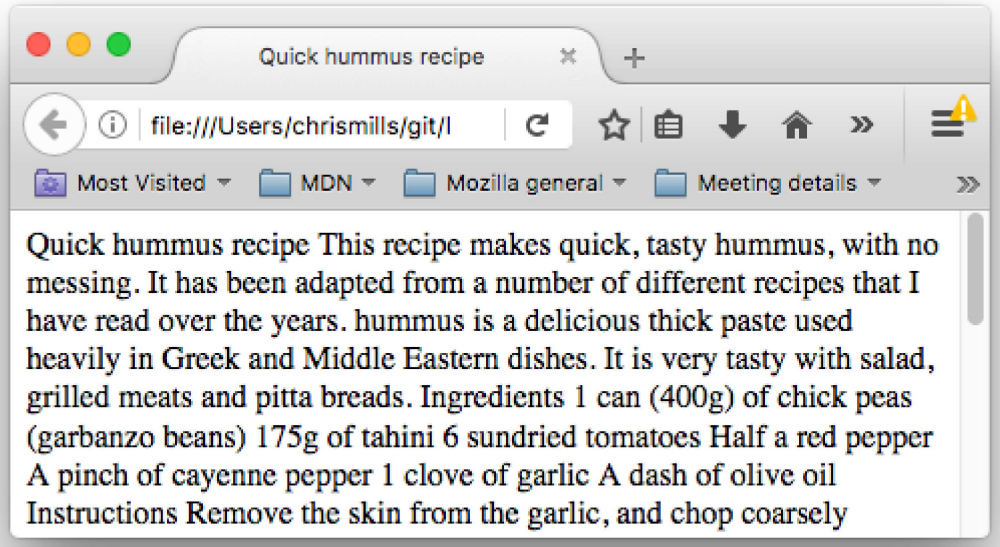
- Preferably, you should use a single `<h1>` per page—this is the top level heading, and all others sit below this in the hierarchy.
- Make sure you use the headings in the correct order in the hierarchy. Don't use `<h3>` elements to represent subheadings, followed by `<h2>` elements to represent sub-subheadings—that doesn't make sense and will lead to weird results.

- Of the six heading levels available, you should aim to use no more than three per page, unless you feel it is necessary. Documents with many levels (i.e., a deep heading hierarchy) become unwieldy and difficult to navigate. On such occasions, it is advisable to spread the content over multiple pages if possible.

Why do we need structure?

To answer this question, let's take a look at text-start.html—the starting point of our running example for this article (a nice hummus recipe). You should save a copy of this file on your local machine, as you'll need it for the exercises later on. This document's body currently contains multiple pieces of content—they aren't marked up in any way, but they are separated with linebreaks (Enter/Return pressed to go onto the next line).

However, when you open the document in your browser, you'll see that the text appears as a big chunk!



This is because there are no elements to give the content structure, so the browser does not know what is a heading and what is a paragraph. Furthermore:

- Users looking at a web page tend to scan quickly to find relevant content, often just reading the headings to begin with. (We usually spend a very short time on a web page.) If they can't see anything useful within a few seconds, they'll likely get frustrated and go somewhere else.
- Search engines indexing your page consider the contents of headings as important keywords for influencing the page's search rankings. Without headings, your page will perform poorly in terms of SEO (Search Engine Optimization).
- Severely visually impaired people often don't read web pages; they listen to them instead. This is done with software called a screen reader. This software provides ways to get fast

access to given text content. Among the various techniques used, they provide an outline of the document by reading out the headings, allowing their users to find the information they need quickly. If headings are not available, they will be forced to listen to the whole document read out loud.

- To style content with CSS, or make it do interesting things with JavaScript, you need to have elements wrapping the relevant content, so CSS/JavaScript can effectively target it.

We therefore need to give our content structural markup.

Active learning: Giving our content structure

Let's jump straight in with a live example. In the example below, add elements to the raw text in the *Input* field so that it appears as a heading and two paragraphs in the *Output* field.

If you make a mistake, you can always reset it using the *Reset* button. If you get stuck, press the *Showsolution* button to see the answer.

Why do we need semantics?

Semantics are relied on everywhere around us—we rely on previous experience to tell us what the function of an everyday object is; when we see something, we know what its function will be. So, for example, we expect a red traffic light to mean "stop," and a green traffic light to mean "go." Things can get tricky very quickly if the wrong semantics are applied. (Do any countries use red to mean "go"? I hope not.)

In a similar vein, we need to make sure we are using the correct elements, giving our content the correct meaning, function, or appearance. In this context the `<h1>` element is also a semantic element, which gives the text it wraps around the role (or meaning) of "a top level heading on your page."

```
<h1>This is a top level heading</h1>
```

By default, the browser will give it a large font size to make it look like a heading (although you could style it to look like anything you wanted using CSS). More importantly, its semantic value will be used in multiple ways, for example by search engines and screen readers (as mentioned above).

On the other hand, you could make any element *look* like a top level heading. Consider the following:

```
<span style="font-size: 32px; margin: 21px 0; display: block;">Is this
```

This is a `<span>` element. It has no semantics. You use it to wrap content when you want to apply CSS to it (or do something to it with JavaScript) without giving it any extra meaning. (You'll find out more about these later on in the course.) We've applied some CSS to it to make it look like a top level heading, but since it has no semantic value, it will not get any of the extra benefits described above. It is a good idea to use the relevant HTML element for the job.

Lists

Now let's turn our attention to lists. Lists are everywhere in life—from your shopping list to the list of directions you subconsciously follow to get to your house every day, to the lists of instructions you are following in these tutorials! Lists are everywhere on the web, too, and we've got three different types to worry about.

Unordered

Unordered lists are used to mark up lists of items for which the order of the items doesn't matter—let's take a shopping list as an example.

```
milk
eggs
bread
hummus
```

Every unordered list starts off with a `<ul>` element—this wraps around all the list items:

```
<ul>
milk
eggs
bread
hummus
</ul>
```

The last step is to wrap each list item in a `<li>` (list item) element:

```
<ul>
  <li>milk</li>
```

```
    <li>eggs</li>
    <li>bread</li>
    <li>hummus</li>
  </ul>
```

Active learning: Marking up an unordered list

Try editing the live sample below to create your very own HTML unordered list.

Ordered

Ordered lists are lists in which the order of the items *does* matter—let's take a set of directions as an example:

```
Drive to the end of the road
Turn right
Go straight across the first two roundabouts
Turn left at the third roundabout
The school is on your right, 300 meters up the road
```

The markup structure is the same as for unordered lists, except that you have to wrap the list items in an `<ol>` element, rather than `<ul>`:

```
<ol>
  <li>Drive to the end of the road</li>
  <li>Turn right</li>
  <li>Go straight across the first two roundabouts</li>
  <li>Turn left at the third roundabout</li>
  <li>The school is on your right, 300 meters up the road</li>
</ol>
```

Active learning: Marking up an ordered list

Try editing the live sample below to create your very own HTML ordered list.

Active learning: Marking up our recipe page

So at this point in the article, you have all the information you need to mark up our recipe page example. You can choose to either save a local copy of our text-start.html starting file and do the work there, or do it in the editable example below. Doing it locally will probably be better, as then you'll get to save the work you are doing, whereas if you fill it in to the editable example, it will be lost the next time you open the page. Both have pros and cons.

If you get stuck, you can always press the *Show solution* button, or check out our text-complete.htmlexample on our github repo.

Nesting lists

It is perfectly ok to nest one list inside another one. You might want to have some sub-bullets sitting below a top-level bullet. Let's take the second list from our recipe example:

```
<ol>
  <li>Remove the skin from the garlic, and chop coarsely.</li>
  <li>Remove all the seeds and stalk from the pepper, and chop coarsely
  <li>Add all the ingredients into a food processor.</li>
  <li>Process all the ingredients into a paste.</li>
  <li>If you want a coarse "chunky" hummus, process it for a short time
  <li>If you want a smooth hummus, process it for a longer time.</li>
</ol>
```

Since the last two bullets are very closely related to the one before them (they read like sub-instructions or choices that fit below that bullet), it might make sense to nest them inside their own unordered list, and put that list inside the current fourth bullet. This would look like so:

```
<ol>
  <li>Remove the skin from the garlic, and chop coarsely.</li>
  <li>Remove all the seeds and stalk from the pepper, and chop coarsely
  <li>Add all the ingredients into a food processor.</li>
  <li>Process all the ingredients into a paste.
    <ul>
      <li>If you want a coarse "chunky" hummus, process it for a short
      <li>If you want a smooth hummus, process it for a longer time.</l
    </ul>
  </li>
</ol>
```

Try going back to the previous active learning example and updating the second list like this.

Emphasis and importance

In human language, we often emphasise certain words to alter the meaning of a sentence, and we often want to mark certain words as important or different in some way. HTML provides various semantic elements to allow us to mark up textual content with such effects, and in this section, we'll look at a few of the most common ones.

Emphasis

When we want to add emphasis in spoken language, we *stress* certain words, subtly altering the meaning of what we are saying. Similarly, in written language we tend to stress words by putting them in italics. For example, the following two sentences have different meanings.

I am glad you weren't late.

I am *glad* you weren't *late*.

The first sentence sounds genuinely relieved that the person wasn't late. In contrast, the second one sounds sarcastic or passive-aggressive, expressing annoyance that the person arrived a bit late.

In HTML we use the `<em>` (emphasis) element to mark up such instances. As well as making the document more interesting to read, these are recognised by screen readers and spoken out in a different tone of voice. Browsers style this as italic by default, but you shouldn't use this tag purely to get italic styling. To do that, you'd use a `<span>` element and some CSS, or perhaps an `<i>` element (see below).

```
<p>I am <em>glad</em> you weren't <em>late</em>.</p>
```

Strong importance

To emphasize important words, we tend to stress them in spoken language and **bold** them in written language. For example:

This liquid is **highly toxic**.

I am counting on you. **Do not** be late!

In HTML we use the `<strong>` (strong importance) element to mark up such instances. As well as making the document more useful, again these are recognized by screen readers and spoken in a different tone of voice. Browsers style this as bold text by default, but you shouldn't use this tag purely to get bold styling. To do that, you'd use a `<span>` element and some CSS, or perhaps a `<b>` element (see below).

```
<p>This liquid is <strong>highly toxic</strong>.</p>

<p>I am counting on you. <strong>Do not</strong> be late!</p>
```

You can nest strong and emphasis inside one another if desired:

```
<p>This liquid is <strong>highly toxic</strong> —
if you drink it, <strong>you may <em>die</em></strong>.</p>
```

Active learning: Let's be important!

In this active learning section, we've provided an editable example. Inside it, we'd like you to try adding emphasis and strong importance to the words you think need them, just to have some practice.

Italic, bold, underline...

The elements we've discussed so far have clearcut associated semantics. The situation with `<b>`, `<i>`, and `<u>` is somewhat more complicated. They came about so people could write bold, italics, or underlined text in an era when CSS was still supported poorly or not at all. Elements like this, which only affect presentation and not semantics, are known as **presentational elements**and should no longer be used, because as we've seen before, semantics is so important to accessibility, SEO, etc.

HTML5 redefined `<b>`, `<i>`, and `<u>` with new, somewhat confusing, semantic roles.

Here's the best rule of thumb: It's likely appropriate to use `<b>`, `<i>`, or `<u>` to convey a meaning traditionally conveyed with bold, italics, or underline, provided there is no more suitable element. However, it always remains critical to keep an accessibility mindset. The concept of italics isn't very helpful to people using screen readers, or to people using a writing system other than the Latin alphabet.

- `<i>` is used to convey a meaning traditionally conveyed by italic: foreign words, taxonomic designation, technical terms, a thought...

- `<b>` is used to convey a meaning traditionally conveyed by bold: key words, product names, lead sentence...
- `<u>` is used to convey a meaning traditionally conveyed by underline: proper name, misspelling...

> A kind warning about underline: **People strongly associate underlining with hyperlinks.**Therefore, on the web, it's best to underline only links. Use the `<u>` element when it's semantically appropriate, but consider using CSS to change the default underline to something more appropriate on the web. The example below illustrates how it can be done.

```
<!-- scientific names -->
<p>
  The Ruby-throated Hummingbird (<i>Archilochus colubris</i>)
  is the most common hummingbird in Eastern North America.
</p>

<!-- foreign words -->
<p>
  The menu was a sea of exotic words like <i lang="uk-latn">vatrushka</
  <i lang="id">nasi goreng</i> and <i lang="fr">soupe à l'oignon</i>.
</p>

<!-- a known misspelling -->
<p>
  Someday I'll learn how to <u style="text-decoration-line: underline;
</p>

<!-- Highlight keywords in a set of instructions -->
<ol>
  <li>
    <b>Slice</b> two pieces of bread off the loaf.
  </li>
  <li>
    <b>Insert</b> a tomato slice and a leaf of
    lettuce between the slices of bread.
  </li>
</ol>
```

Test your skills!

You've reached the end of this article, but can you remember the most important information? You can find some further tests to verify that you've retained this information before you move on—see Test your skills: HTML text basics.

Summary

That's it for now! This article should have given you a good idea of how to start marking up text in HTML, and introduced you to some of the most important elements in this area. There are a lot more semantic elements to cover in this area, and we'll look at a lot more in our Advanced text formatting article, later on in the course. In the next article, we'll be looking in detail at how to create hyperlinks, possibly the most important element on the web.