Semantic Elements

The Problems with Only Using `div`

Using the `<div>` element in itself is not an issue. It serves a purpose and there's no reason not to use it.

But using DIV exclusively in your HTML can cause issues for you and anyone else who works on your project.

Readability Issues

If you have ever looked at someone else's code, or even your own code months after writing it, it can be hard to scan if it's populated solely with `<div>` elements.

Deciphering the layout can take much longer than necessary, which is like Kryptonite for your productivity. Just trying to figure out where the closing `</div>` tag is for a specific block of code can be tedious.

Accessibility Issues

Adhering to a11y considerations is not just about color, contrast and captioning. The World Health Organization estimates that 285 million people are visually impaired worldwide: 39 million are blind and 246 million have low vision.

This is another reason why it's important to write your HTML to be as accessible as possible, which means writing semantic code.

Screen readers need context in order to accurately read a web page out loud. To a screen reader, elements like `<div>` and `<span>` mean nothing. Semantic divs like `<form>` and `<button>` are easier to parse.

Consistency Issues

If you know what to expect when working on a project with a team, you'll be much more effective. You'll also tend to have fewer bugs in your code.

Setting a standard for using semantic HTML makes sense because anyone picking up the project will more easily understand the layout.

Additionally, when you start applying or tweaking the CSS for an HTML document, you'll find it much faster and easier to find specific elements when semantic HTML was used in the layout.

SEO Issues

When using semantic markup, search engines will consider its contents as important keywords to influence the page's search rankings.

What is semantic HTML?

The clearest definition of semantic HTML that I found states that:

> A semantic element clearly describes its meaning to both the browser and the developer.

Using semantic HTML is like the difference between pointing out an object in the sky and saying, "Look, an object!" or "Look, an airplane!"

In fact, using semantic HTML is noted as part of the HTML5 standard:

Authors are strongly encouraged to view the

For example, which is easier to scan:

```
<div class="quote" id="twain-quote">
  "Get your facts first, then you can distort them as you please." – Mark Twain
</div>
```

Or

```
<blockquote>
  "Get your facts first, then you can distort them as you please." – Mark Twain
</blockquote>
```

In the second example, you can see the `<blockquote>` element, which is immediately understandable as text that needs to be displayed in a quotation format.

Using `div` alternatives may take a little more thought, but that bit of extra planning with semantic HTML will be worth it in the end.

Alternatives to `div` in HTML

Let’s talk about some of the more common `div` alternatives. Chances are you’ve seen these elements before, but here we’ll cover exactly what they’re for and how to use them.

The <nav> element

```
<nav>
  <ul>
    <li>
      <a href="/">Home</a>
    </li>
    <li>
      <a href="/">About</a>
    </li>
    <li>
      <a href="/">Contact</a>
    </li>
  </ul>
</nav>
```

The nav element is exactly what it sounds like. You use this element to outline a set of navigation links.

As mentioned before, this also allows screen readers to decide whether or not to display this type of content initially. The `nav` element is best used for a major block of navigation links in the document.

The <main> element

```
<main>
  <h1>The Godfather of All Content</h1>
  <h2>The Wedding</h2>
  <p>
    Why did you go to the police? Why didn't you come to me first? Vito, how do
    you like my little angel? Isn't she beautiful? Only don't tell me you're
    innocent. Because it insults my intelligence and makes me very angry. I see
    you took the name of the town. What was your father's name? The hotel, the
    casino. The Corleone Family wants to buy you out.
  </p>
</main>
```

Similar to `<nav>`, the main element is used just as it sounds (semantics at work again). This element wraps the blocks of code that specify the main content of the page, or document. The main element will live between the opening and closing `<body>` tags.

The <section> element

```
<section>
  <h1>The Best Sandwich Ever</h1>
  <p>
    The best sandwich is a mutton, lettuce and tomato, where the mutton is nice
    and lean. It's so perky, I love that.
  </p>
</section>
```

The `<section>` element is a great example of using a `div` alternative to separate content.

In the example above, we are separating an introduction and opening paragraph into two sections. Finding and styling those sections in our CSS document will be much faster than hunting down a `<div>` class.

The <header> element

```
<header><img src="/" id="logo"></header>
```

The `<header>` element is different from the `<head>` element in that you can use it multiple times throughout the document.

For example, you could use one set of `<header>` elements to place a logo and another set to describe a header for a specific piece of content, like an article (more on this later).

The <footer> element

```
<footer>
  <p>© 2021 All rights reserved. Don't steal.</p>
  <p>Contact: <a href="mailto:jiffy@jiffysites.com">Email Jiffy!</a></p>
</footer>
```

Just like with the `<header>` element, you can use `<footer>` elements anywhere in your HTML document.

A typical use for `<footer>` is for copyright or author information. You can also use a footer element as a closing within a `<section>` element.

Less common alternatives to `div`

There are also some elements you may not have seen before or have only seen very rarely. But they do come in handy and learning them will help make your code much more readable.

The <aside> element

```
<p>
  My favorite TV show of all time is The Muppet Show. It's sweet, funny and
  brilliant.
</p>
<aside>
  <h3>The Muppet Show</h3>
<p>The Muppet Show was created by Jim Henson and aired from 1976 – 1981.</p>
</aside>
```

In film or theater, an aside is known as a dramatic device in which a character speaks to the audience, separately from the main dialogue.

That's exactly how we can use the <aside> element in our HTML. We are making a notation that's related to the content, but that we want to keep separate. We can use it in a sidebar as well.

The <code> element

```
<p>
  You can use this for a piece of code such as:
  <code class="gray-code">const muppetFrog = 'Kermit'</code>, which looks
  different from the other text.
</p>
```

In instances where we want to differentiate a piece of code from regular text, the `<code>` element comes in very handy. The resulting text when rendered in the browser might look like this, with a little CSS styling:

You can use this for a piece of code such as: `const muppetFrog = 'Kermit'`, which looks different from the other text.

The <article> element

```
<article class="all-muppets">
  <h1>Muppets</h1>
  <article class="kermit">
    <p>Kermit is the Muppet leader.</p>
  </article>
  <article class="fozzy"><p>Fozzy is a stand-up bear.</p>
  </article>
  <article class="piggy"><p>Don't mess with Miss Piggy.</p>
  </article>
</article>
```

The `<article>` element is yet another element that's used to differentiate types of content from one another in the layout.

An `<article>` should be self-contained content that's separate from the main content. Using this element also makes it easier to style with CSS, making the article content clearly different from other content on the page.

The <blockquote> element

The `<blockquote>` element is another straightforward one, used just as it sounds: to separate a quotation from other text, as seen earlier.

The <mark> element

```
<p>
  This is a sentence about the best Muppet ever, which happens to be
  <mark>Pepe the King Prawn</mark>.
</p>
```

The `<mark>` element is not only a perfect way to highlight a block of text, it's easily understood when you come across it in a document.