HTML5 Semantic Elements

(https://github.com/learn-co-curriculum/phase-1-html5-semantic-elements) (https://github.com/learn-co-curriculum/phase-1-html5-semantic-elements/issues/new)

Learning Goals

- Explain the historical reasoning behind semantic elements
- Demonstrate HTML5 semantic element use

Introduction

We have `div` elements to organize and label sections of HTML. But too many `div` tags makes our HTML document look cluttered and confusing. Even when we identify or classify them with specific names, it's a lot to keep track of. It would be clearer for both developers and browsers if we could use more descriptive HTML elements to arrange our content. That's where semantic elements come in. They allow us to say, "this isn't *just* a division of text, it's a *header*." We'll learn more about these *semantic* elements in this lesson.

Explain the Historical Reasoning Behind Semantic Elements

When developers first began defining containers to structure HTML, they had only one generic element available to them: the `div`. Creating complex page layouts then required dozens of `div` elements that were often difficult to organize or locate within the code. HTML authors needed a way to distinguish one `div` from another, which led to `id` and `class` attributes on elements being misused in an attempt to communicate what the `div` was doing. For example, a document usually has only one header, so it seemed sensible to write `div id="header"` as a way to say something stronger than, "this is a text division." Developers wanted to say, "this is a special division of introduction," but lacked the specific language to express it. They wanted those sections to have a *semantic meaning* (more on that later). But the HTML language simply didn't have the ability to meet this need.

Nevertheless, many HTML authors thought that this was a good idea and an informal standard sprang up around adding `id` attributes on elements to express their "semantic meaning."

```
<div id="header">
  <div class="wrapper">...</div>
</div>
```

When the W3C (the organization that oversees the specifications for HTML and CSS) started writing the specification for HTML5 they wanted to create new elements that would eliminate the need to label so many `div` elements. The goals were to make the code more readable for developers and more descriptive for browsers. It turned out that many developers were already using the same names to label their elements, such as `id="header"` , `id="footer"` , `id="nav"` , `class="article"` , etc. So HTML5 provided semantic elements that explicitly described those functions for developers to use instead.

We once used to have to identify a `div` as our header section.

```
<div id="header">...</div>
```

Now we use the `header` element.

```
<header></header>
```

Why do we call these *semantic* elements? Semantic elements are elements that we use when the content within the element all has the same related *meaning*. In our `header` example above, all the content we would put within the `header` element would relate to introductory content, such as titles or navigation.

Demonstrate HTML5 Semantic Element Use

Let's take a layout that uses `div` elements and convert it to use semantic elements instead. This is the markup we begin with:

```
<div class="wrapper">
  <div id="header">
    <div id="nav">...</div>
  </div>
  <div id="main">
    <div id="music">
```

```
      <div id="rock">...</div>
      <div id="jazz">...</div>
    </div>
  </div>
  <div id="aside">...</div>
  <div id="footer">...</div>
</div>
```

Now we'll replace each instance of a `div` with a semantic element that matches the type of content we want it to contain.

```
<div class="wrapper">
  <header>
    <nav>...</nav>
  </header>
  <main>
    <section id="music">
      <article id="rock">...</article>
      <article id="jazz">...</article>
    </section>
  </main>
  <aside>...</aside>
  <footer>...</footer>
</div>
```

Notice that in cases where the content within the element is not semantically related or we have the need to create a generic box such as the `wrapper`, we can still use `div` elements as we please. As a developer, keep in mind that while these elements are intended for certain content, there are no hard rules about how to configure them. You should feel comfortable configuring them in any way that makes the most sense to you and best suits your layout needs.

Conclusion

If the content within an element is all semantically related, it is best practice to use the appropriate HTML5 semantic element if one applies. This cleans up our code and makes it more readable for developers and more descriptive to browsers. There are a variety of semantic elements you can use to structure your content, and you can also still use `div` elements to create generic boxes or grouping elements as needed.

Resources

- **Dive into HTML5 - Semantic Elements (http://diveintohtml5.info/semantics.html#new-elements)**
- **MDN - HTML - Element Reference (https://developer.mozilla.org/en-US/docs/Web/HTML/Element)**