HTML5 - Intro

- Dr Nick Hayward

A brief overview of HTML5.

Contents

Intro

HTML5 was finally standardised in October 2014, after many years of effective usage, modifications, and suggestions.

HTML5 introduces some interesting new features, which allow web designers to create powerful, semantically structured web sites.

For example,

- a new canvas element has been added for drawing
- video and audio support is now available as an embedded feature of a web page. Before, it required a plugin such as Flash to enable playback features.
- improved support for local offline storage
- new content specific elements such as
  - article, footer, header, nav, section
- new form controls such as
  - calendar, date, time, email, url, search

There are also new input type attribute values. These have been designed to provide better input control before the form is sent to the server side.

We can check browser compatibility using the following tool,

- HTML5 Test

Doctype

The `DOCTYPE` is a special instruction to the web browser concerning the required processing mode for rendering the document's HTML.

The `doctype` is a required part of the HTML document, and is the first part of our HTML document. As such, it should always be included at the top of a HTML document. For example,

```
<!DOCTYPE html>
```

or

```
<!doctype html>
```

This is the doctype we add for HTML5 rendering. Therefore, this is not a HTML element, it simply tells the browser the required version of HTML for rendering the current page.

Basic template

Every HTML5 page we may design needs to begin with the following basic template. It will include an initial, basic DOCTYPE

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title></title>
  </head>
  <body>

  </body>
</html>
```

which is a notable contrast with the older DOCTYPE of HTML4 and XHTML, and thankfully a lot simpler as well. We may initially set a broad charset for our document as well using Unicode's `utf-8` value.

We may use this simple template as our initial shell for beginning a project's code with HTML5.

n.b. UTF-8 is a character encoding capable of encoding all possible characters, or code points, defined by Unicode and originally designed by Ken Thompson and Rob Pike. The encoding is variable-length and uses 8-bit code units.

Elements

Often known simply as *tags*, elements allow us to add a form of metadata to our HTML page. For example,

```
<!-- a paragraph element -->
<p>add some paragraph content...</p>
<!-- a first heading element -->
<h1>our first heading</h1>
```

This metadata is used to allow us to apply structure to a page's content.

add some extra structure

For example, we may wish to add some extra structure to our new template.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <!-- title for the webpage appears in the window, tab heading... -->
    <title>Demo 1</title>
  </head>
  <body>
    <h1>Our first web page</h1>
    <p>
      As we build our web apps, more elements and content will be added...
    </p>
  </body>
</html>
```

It's still very basic, but we now have a web page that will load and render a heading followed by some text content in a paragraph. You can also see that we've added a title for our new page.

add some comments

There's also a simple comment, to help us understand the page. Comments in HTML5 may still be written as follows,

```
<!-- a comment in html -->
```

The comment will not be visible to the user in the browser, it appears in the underlying source code for reference purposes, documentation, and so on.

Semantic elements

HTML5 also adds some new semantic elements. They include the following elements,

```
<article>
<aside>
<details>
<figure>
<figcaption>
<footer>
<header>
<main>
<nav>
<section>
```

These new elements allow us to better structure our underlying documents, adding clear semantic divisions as well. It's doubtful you'll use all of these elements in every page you build, but some will become firm favourites over time.

a basic example

For a bit of fun, and to ensure we test most of the above new elements, we might render the following example

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <!-- our second demo with lots of new elements -->
    <title>Demo 2</title>
  </head>
  <body>
    <header>
      <h1>Our first web page</h1>
    </header>
    <!-- navigation elements, links... -->
    <nav>Option 1</nav>
    <!-- main content -->
    <main>
      <section>
        <p>
          As we build our web apps, more elements and content will be added...
        </p>
        <figure>
          <img src="media/images/philae-demo2.jpg" alt="temple of philae"
width="333px" height="200px">
        </figure>
      </section>
      <aside>
        Temple at Philae in Egypt is Ptolemaic era of Egyptian history...
      </aside>
    </main>
    <footer>
      foot of the page...
    </footer>
  </body>
</html>
```

Most of the above tags should be self-explanatory. However, `aside` normally requires some extra definition. The `aside` element tag is used to define some content aside from the content which contains this element. It is normally used to help define or relate material to this surrounding content. It acts, effectively, as supporting or additional contextual material for this surrounding content. It may be used, for example, to add additional material and content to a sidebar.

avoiding confusion with semantic elements

You should notice from this demo that we have not used the element tag `article`. This, plus the tag `section`, can still cause some confusion amongst web developers.

They are not as widely used as you might expect, often because it is simply easier, and more practical, to use the traditional `div` element, even though it is often described as the sectioning element of last resort. A sort of **catch-all** option for elements and sectioning.

The best analogy is with a standard newspaper. Each newspaper will contain numerous different sections, such as headlines, politics, health, sports etc, and within each section we will also find articles. However, according to the HTML5 specification, an article element also

> > represents a self-contained composition in a document, page, application, or site and that is, in principle, independently distributable or reusable, e.g. in syndication.

One of the issues, and often noted concerns, with using semantic elements is how and when to add them to our document. In effect, where and when do we add them to our page?

Using the more traditional non-semantic elements was considered simpler due to their generalised application within a page's structure.

So, let's go through some of these new elements and where to add them to our web page.

`<header>` and `<nav>` elements

The `<header>` element is used to collect and contain introductory content, which is semantically appropriate for the head or top of a page.

It is technically feasible and acceptable to include multiple `<header>` elements within a page or document. For example, we might include a header for the main content, the sidebar content, an article, and a section. However, it shouldn't be used as a sectioning element in its own right throughout a document.

The `<nav>` element is short for navigation and, as you might expect, stores and defines a set of links for internal or external navigation. As with the header, this tag is not meant to define all navigation links, but instead can often be considered suitable for primary site links.

It is common practice, for example, to add additional links to a sidebar, footer or the main content of a page. However, there is no need to consider these within a `<nav>` element structure. They are, effectively, additional, often internal, links for a site. Links in the site's footer are a good example.

`<main>` element

Simply put, this element tag defines our **main** or primary content, traditionally the central content area of our page or document. For those who have developed or used earlier HTML standards, such as HTML4, often we simply used a `<div>` element with a `class` or `id` to define this section of the page as the central content. For example,

```
<!-- e.g. HTML4 main content -->
<div id="main">
  ...
</div>
```

However, with the advent of HTML5 we can semantically mark our pages to define our **main** content.

As a note, this element tag should not include any page features, such as navigation links, headers etc, that are repeated across multiple pages.

Also, we cannot add multiple `<main>` elements to a single page. And, it must not be structured as a child element to

```
<article>, <aside>, <footer>, <header> or <nav>
```

`<section>`, `<article>`, `<aside>` elements

As you might imagine from the tag name, this element defines a section of a page or document.

Therefore, the W3C defines this tag as follows,

> > a section is a thematic grouping of content. The theme of each section should be identified, typically by including a heading as a child of the section element.

Source - W3C Documentation

A site can, therefore, be sub-divided into multiple **sections** as we might traditionally consider a book, for example.

The `<article>` element, again as with traditional print, may be considered suitable for organising and containing independent content. We may include multiple articles, as long as the disparate organisation of material is logical and required.

So, we might consider using the `<article>` element for a post of some kind, such as a forum or blog post, a story, a newspaper report or article, a review of content or a product, and so on. The key to using this tag is to consider whether the material can be used in isolation. i.e. can we separate the content for syndication.

As mentioned above, the `aside` element tag is used to define some content aside from the content which contains this element. It is normally used to help define or relate material to this surrounding content. It acts, effectively, as supporting or additional contextual material for this surrounding content.

To help us better understand how to use and consider these elements, MDN (Mozilla Developer Network) Documentation has a great summary as follows,

- if it makes sense to separately syndicate the content of a `<section>` element, use an `<article>` element instead
- do not use the `<section>` element as a generic container; this is what `<div>` is for, especially when the sectioning is only for styling purposes. A rule of thumb is that a section should logically appear in the outline of a document.
- MDN Documentation suggests,

> > if it makes sense to separately syndicate the content of a `<section>` element, use an `<article>` element instead

and

> > do not use the `<section>` element as a generic container; this is what `<div>` is for, especially when the sectioning is only for styling purposes. A rule of thumb is that a section should logically appear in the outline of a document.

`<figure>`, `<figcaption>` elements

Again, as with print media, we may organise our image content along traditional lines with a semantically logical grouping of image and caption. This grouping is the `<figure>` element, which acts as a parent group for image and caption. For example,

```
<figure>
  <img src="media/images/philae-demo2.jpg" alt="temple of philae"
  width="333px" height="200px">
  <figcaption>Ptolemaic temple at Philae, Egypt</figcaption>
</figure>
```

So, our previous example can now be updated with the simple addition of a useful figure caption.

`<footer>` element

As the element tag might suggest, this defines a footer for a document or section. A `<footer>` element usually contains information about its containing element. So, let's consider a couple of examples.

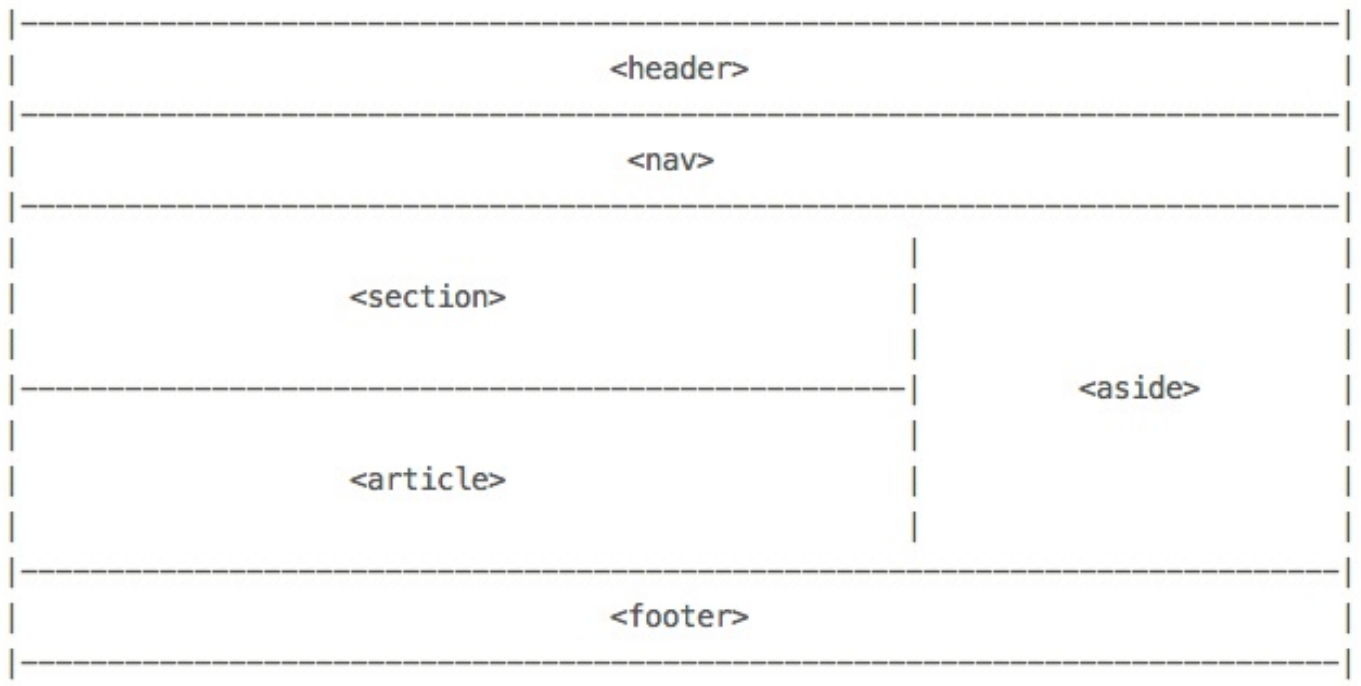
1. in a footer for an article, we might use this element to define and record the author of the article, publication date, any suitable tags or metadata, associated documents, and so on.
2. in its primary use, a footer will simply be placed at the **foot** of a page, where we may record copyright information, contextual links, contact information, small logos, and so on. This has traditionally been the semantic usage of a footer within a page. We would normally use a `<div>` with a `class` or `id` value set to `"footer"` for HTML4 and earlier.

Example page structure

With our brief knowledge of HTML5 semantic elements, we may now consider how they fit together within our page puzzle.

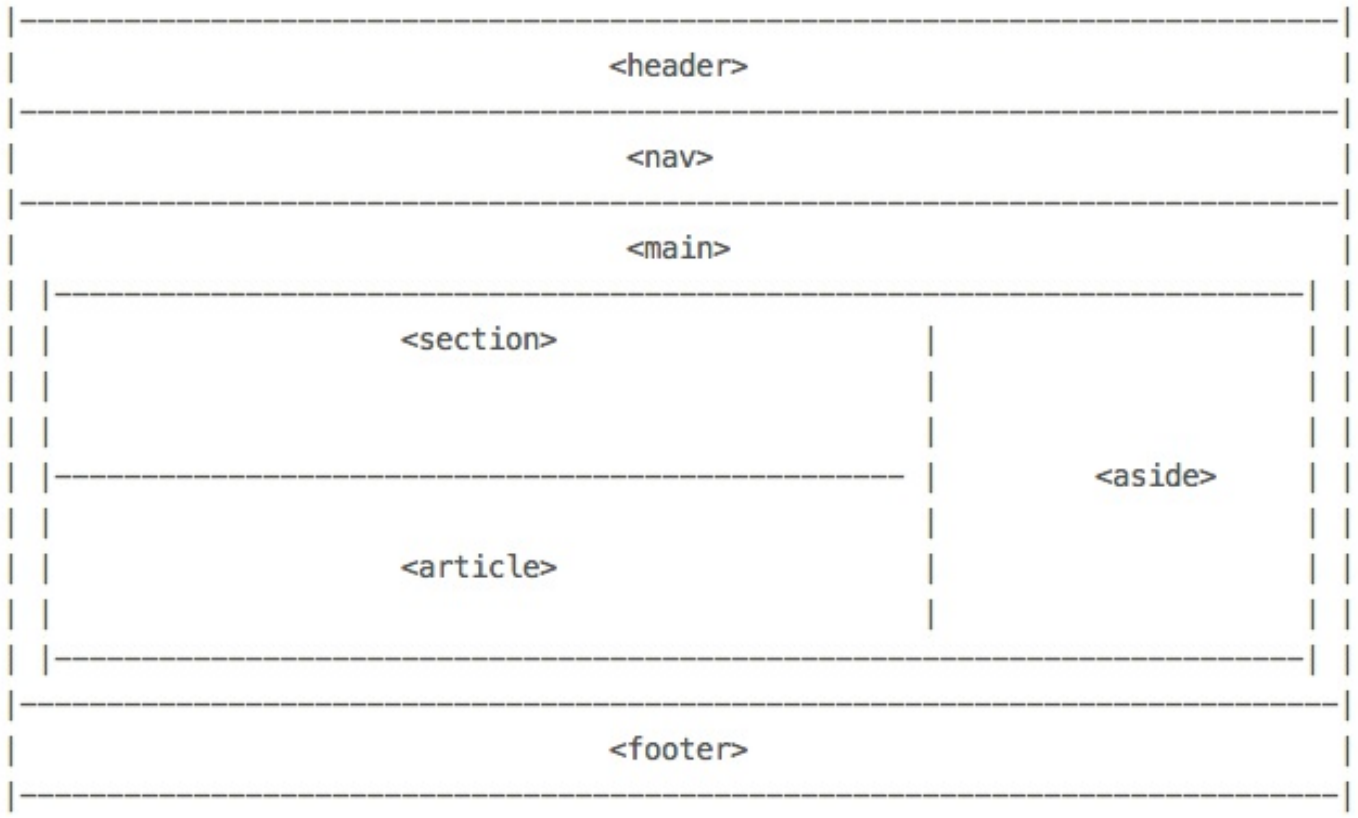
For example, the following image shows a basic page structure

Image - Example 1



We can also add further semantic division with the logical addition of the `<main>` HTML5 element. For example,

Image - Example 2



You should notice that we have not included the `<html>` and `<body>` tags to these diagrams. They are required for all HTML documents, so we can safely avoid replicating their usage in multiple diagrams.

We have now divided the page into four logical, semantic divisions

- header
- nav
- main
- footer

We could also add a sidebar to help us divide the page's content. There are many different options available for page organisation, but the basic template will often be as we've just seen.

References

- HTML5 Test
- MDN
  - HTML developer guide
  - Block-level elements
  - Content categories
  - Inline elements
- W3C
  - HTML5 Documentation
- W3 Schools
  - W3Schools - HTML5 Semantic Elements