HTML Semantic Layout Tags

HTML5 includes new tags to be used for layouts:

- header
- footer
- main
- nav
- article
- aside
- section

These new tags were added to better describe the page content according to the content type. These tags have **semantic** meaning while the previous use of `<div>` tags did not have any particular meaning attached. Previously, although the `id` for the `<div>` *described* the content, the `id` did not have any effect on screen readers while semantic tags ensured the content is correctly classified. The issue with divs is that because there is no attached semantic meaning, you cannot tell whether the div is used for structuring or is just used for styling (i.e. as a block element). Note that there were more semantic layout tags added other than the above seven elements, but the above listed are the most essential ones.

semantic (si'mantik)
adj. relating to meaning in language or logic
(Oxford Dictionary)

When we speak of HTML being "semantically correct," it means that the HTML is correctly structured and described based on the content's meaning.

Note

As mentioned, these tags are new to HTML5. There should be no worries in using them, however. In older browsers, these unknown tags are treated as divs.

Block-, inline-, and section-level elements

Block-level elements

Some elements are, by default, displayed on a line of their own on a page (or in its container element). For example, a paragraph by default always takes up the whole width of its containing element. These are called block elements because they take up a whole **block** of space.

Inline-level elements

Some other elements are, by default, displayed *next to* other elements as if in a line. These are called inline elements. Some examples of this type of element are <span>, <a>, <strong>, <em>.

Block elements CANNOT be children of inline elements! One exception is wrapping <a> around a <div>. Under older standards, this was frowned upon but as it is a common usage pattern, it is now acceptable.

Section-level elements

The new HTML5 layout tags listed above are all section-level elements. Section elements are also block elements but have the express purpose of defining sections of content or sections of the page. Section elements **must** have headings. For generic block elements which are not *sections*, use `<div>`.

<main>

This element defines the main content of the page. For this reason, there can only be ONE `<main>` element in an HTML page. You cannot have more than one piece of main content.

The `<main>` element cannot be a descendant of an `<article>`, `<aside>`, `<footer>`, `<header>`, or `<nav>` element.

<article>

Articles are self-contained compositions. These should be used for news articles, blog posts, forum posts, etc. In a blog, where there is a listing of blog posts, each of those posts can be placed within an `<article>` element. Note that if you have only one `<article>` as the child of `<main>`, this is redundant because that means that the article itself is the main content. For this case, you can simply just place the article **content** inside `<main>` and forego the `<article>`.

<aside>

An `<aside>` element defines a section of the page where the content is kind of related to the content at the same level as the aside but is separate from that content. You can use asides for sidebars. In an article about Europe, you can have an aside containing information about France. This is separate since the main article is about Europe. (In books, asides would be the extra information in a box to the side of the main text. You find these often in non-fiction books such as a text book.)

<header>

Usually, this element is used to define the page header. This can also be used to define the header of a `<section>`, `<article>`, `<aside>`, and `<nav>`.

<footer>

A `<footer>` defines the footer for its nearest *ancestor* sectioning type of content (e.g. `<section>`, `<main>`, `<article>`, `<aside>`, `<nav>`, `<body>`). Usually footers contain author info, copyright data, footnotes, related links, etc. Footers containing sections are usually used for appendices, legal information, and so on.

<nav>

Nav elements are used to define a section with navigation links (e.g. the main menu or a side menu). In a menu, links are placed in a list. Typically, you could have an aria-label to label what the nav is (e.g. `aria-label="Main navigation"`).

<section>

Sections are used to define **generic sections** of a page (i.e. not a header, footer, aside, nav or main section). Sections group logical pieces of content together. Sections should not be used to define generic *containers* where you just need to wrap a piece of content, say to make it a block or just to have a wrapper container (e.g. to wrap a video). For those cases, you should use a div instead. Each section potentially has a heading and its own outline.

Sections usually are identified with a heading. Because sections are their own grouping, headings should ideally start at H1 again, though this is not recommended in practice (see the note below about accessibility). Remember to use H2, H3, H4, etc. appropriately to properly define the structure of your section. (Prior to HTML5, because there was no `<section>`, there could only be one H1 per page. In HTML5, sections define a separate level, so you can technically have an H1 in a section even if there is already an H1 in the main content area.)

According to W3C, however, even though the HTML5 spec says that you can put H1 within a `<section>`, some screen readers still follow the older H1 meaning of "top-level heading". (This is why **for accessibility, it is best to use ONE <h1> per page**.) One suggestion is to continue using H2 and H3 accordingly even though it is the first heading within a section.

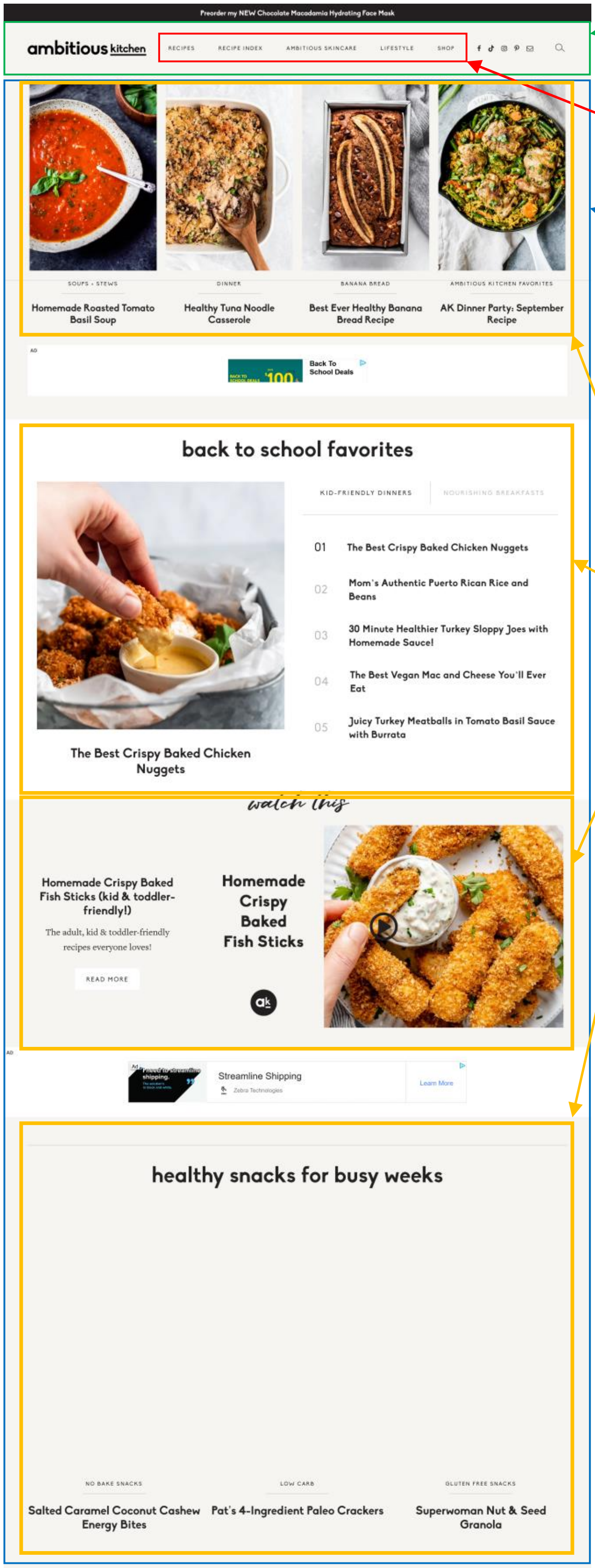
Tip!

To check your outline structure, you can use an HTML5 outliner tool to quickly check if your structure is nested correctly. One such tool is found at *https://gsnedders.html5.org/outliner/*.

Practice

Let's take a look at the home page (partial) of Ambitious Kitchen. Only a partial is shown to keep it short.

Identify the page parts (see the next page):



Homemade Roasted Tomato Basil Soup

Healthy Tuna Noodle Casserole

Best Ever Healthy Banana Bread Recipe

AK Dinner Party: September Recipe

back to school favorites

The Best Crispy Baked Chicken Nuggets

01 The Best Crispy Baked Chicken Nuggets

02 Mom's Authentic Puerto Rican Rice and Beans

03 30 Minute Healthier Turkey Sloppy Joes with Homemade Sauce!

04 The Best Vegan Mac and Cheese You'll Ever Eat

05 Juicy Turkey Meatballs in Tomato Basil Sauce with Burrata

Homemade Crispy Baked Fish Sticks (kid & toddler-friendly!)

The adult, kid & toddler-friendly recipes everyone loves!

Homemade Crispy Baked Fish Sticks

healthy snacks for busy weeks

Salted Caramel Coconut Cashew Energy Bites

Pat's 4-Ingredient Paleo Crackers

Superwoman Nut & Seed Granola

Site <header>: this usually holds the site name and main menu for the site.

<nav>: Note that you should only use <nav> for **major** site navigation.

<main>: On a home page, usually all of the content that's not the site header or footer is featured content and so serves as the main content for the home page. This may not always be the case as it depends on the content.

On a secondary, non-home page (e.g. an article page), the actual content (the article—i.e. the stuff you went on the page to read) is the main content.

<section>: This should be used for page sections which are distinct. Sections are content with a title and other content so don't just use it for styling purposes (that's what <div> is for). In the page to the left, each outlined section is clearly distinct (i.e. different types of content) from each other with headings identifying what type of content it is.

Although sections are sometimes used to separate article sections, I would avoid this as proper subheading usage would already imply there's a subsection in the article. Over-usage of <section> can clutter up the semantics.

It should be used when you are defining some sort of section but other section-level are not applicable (i.e. when <nav>, <header>, <footer>, <main>, <article> don't make sense.