30 Best HTML5 Practices 2018

Editoral Staff

HTML5 has been around for many years now and has been stable and supported at least partially supported by most major browsers since 2014. We are going to present here a list of best coding practices regarding HTML5 markup.

This list is from a perspective of creating a clean, maintainable and scalable code, that will make a good use of the semantic markup elements of HTML5 and that will render correctly in supported browsers. Best SEO practices, CSS and js scripting practices and generic front-end development practices are beyond the scope of this article.

A. General

01 – Declare a doctype

The DOCTYPE declaration should be in the first line of your HTML. It is recommended that you use the HTML5 doctype:

```
<!DOCTYPE html>
```

which actually activates the standard mode in all browsers. Alternatively, you can use a doctype that corresponds to the HTML/XHTML version you are using.

02 – Closing tags

- Void elements (tags that cannot have any contents)
  Self-closing tags are valid, but not required. These elements include:
  `<br>`, `<hr>`, `<img>`, `<input>`, `<link>`, `<meta>`,
  `<area>`, `<base>`, `<col>`, `<command>`, `<embed>`, `<keygen>`, `<param>`,
  `<source>`, `<track>`, `<wbr>`

- Normal elements can never have self closing tags.

03 – Optional tags

Some tags are optional in HTML5, because the element is implied to be present. For example, even if you omit the `<html>` tag im the markup, it is implied that your markup is enclosed in an `<html>` element. Other optional tags are `<head>`, `<body>`. Also for some elements, only the closing tag is optional (see below)

NOTE

Optional closing tags
HTML5 considers optional the end tags for several elements. You are not required to use ,
The Google Style Guide for HTML recommends omitting all optional tags.

However this practice has not been widely adopted and, taken out of context, it can be a bit misleading: W3C actually details under what conditions a start or end tag becomes optional –
please see here for more details

04 – The “lang” attribute

One reason for sticking with the use of optional tags, such as the `<html>` tag, is the use of attributes. It is considered best practice for internationalization purposes, according to W3C, to always declare the default text language of a page in the `<html>` tag.

05) “Keep it simple” principle:

Generally, HTML5 was designed for backwards compatibility with older HTML versions and XHTML. For that reason, it is recommended to avoid using XML declarations or attributes.

There’s no reason for this:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!DOCTYPE html>
```

unless you really want to write an XHTML document
Similarly, you don't need xml attributes, such as:

```
<p lang="en" xml:lang="en">...</p>
```

B. Metadata

06 – The `<base>` tag

This is a very useful tag, especially for developing in local servers. However, it has some non-intuitive behaviors when misused. In effect, if you declare a
`<base href="http://www.example.com/" />`
tag, then every link in the document will be relative, unless explicitly specified. This changes the default behavior of some links.
For example, an internal link
`href="#internal"`
will be interpreted as
`href="http://www.example.com/#internal"`

Also, linking an external webpage in this way:
`href="example.org"`
will be interpreted as
`href="http://www.example.com/example.org"`

It is safer to always use absolute paths for your links

07 – The `<title>` tag

It should never been omitted. Besides the obvious fact that the title of your document is not rendered on the browser tab, it is bad practice for accessibility.

08 – Declare the character encoding

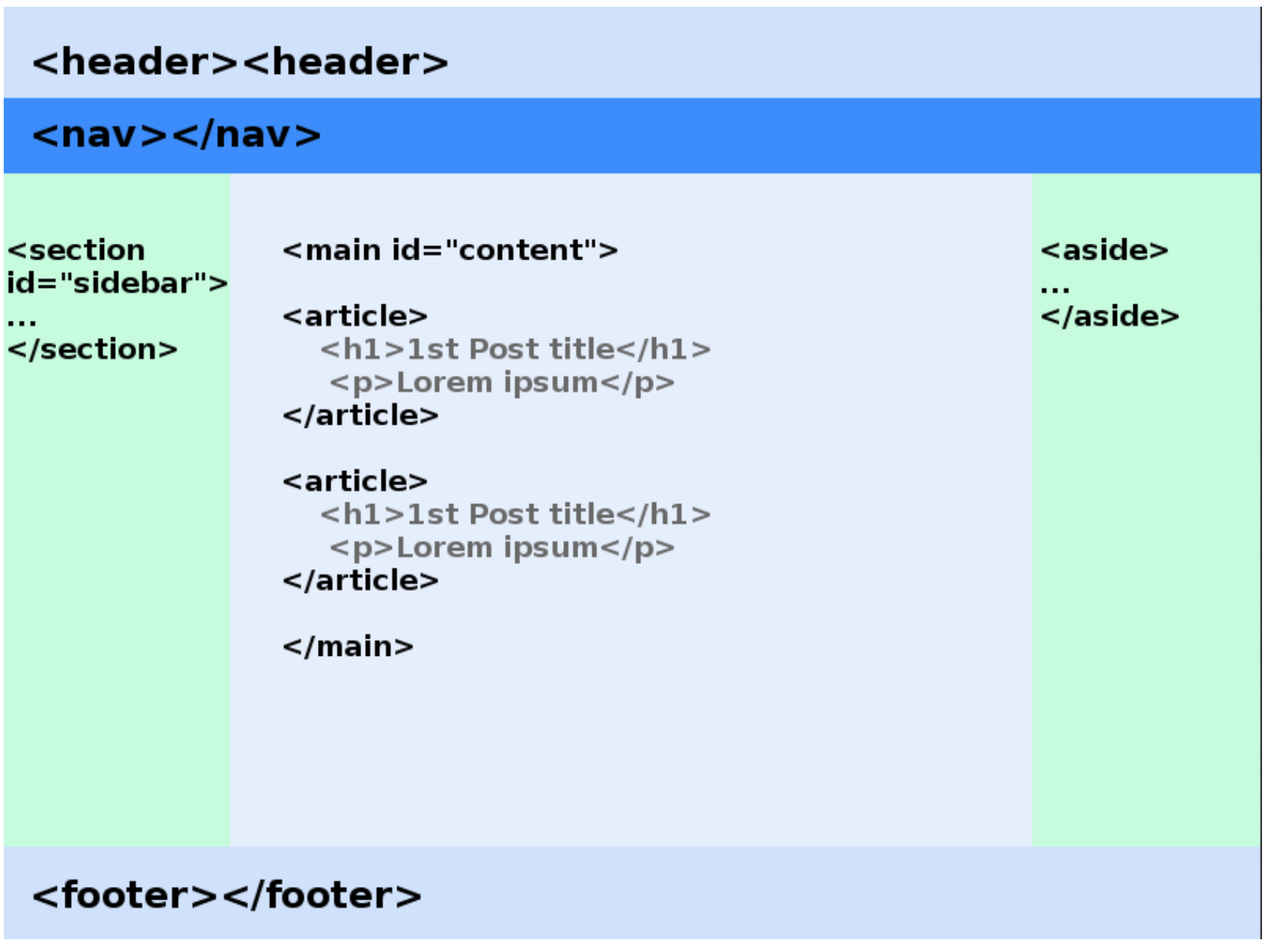
Do not forget the `<meta charset='utf-8'>` (or the declaration of the character encoding used in your document) – it will ensure that your page is always viewed correctly.

09 – Description metatag

This is not strictly a part of HTML best practices, but it is worth to be noted. The `<meta name="description">` attribute is what crawlers and search engines pull when they index your page – if it is present, it will appear as your site description.

C. Use semantically appropriate elements to layout your page

HTML5 offers several elements that will help you organize your layout in appropriate sections.



10 – The `<header>` and `<footer>` elements

In the above picture of a sample layout, we see a `<header>` on the top of the page and a `<footer>` on the bottom. This reflects the typical webpage we are used to see, with a logotype on the top of the page, and the footer with some links and copyright notices on the bottom. WordPress users might be accustomed to name this “masthead” and “colophon” respectively.

However, HTML5 gives a more semantic meaning to the header and footer elements. A `<header>` tag can be used in any section or article to include headings, publish date or other introductory content of the article or section. Similarly, the footer can include information about the author of each article, links to related content, etc.

11 – The `<nav>` element.

The nav element should be used for site-wide navigation. There is no need to declare a role, like this:

```
<nav role="navigation"></nav>
```

The role is implied in the tag:

```
<nav></nav>
```

12 – The `<main>` element

The main element has been included in HTML5 and HTML5.1 specifications to denote the main content of the document body, a content area that is directly related with the major topic of the document
So, there is no need anymore to use a div:

```
<div id="content"></div>
```

when we have a more specific tag for our main content:

```
<main id="content"></main>
```

13 – `<article>`, `<section>`, or `<div>`?

We use `<article>` for a content block that is stand-alone and makes sense without the need to give further context.
A `<section>` tag is used to divide either the page into different subject areas, or to section an individual article. We could say that `<section>` is a more generic tag than `<article>` – it denotes content that is related, but not necessarily self-contained, while an article always has the stand-alone property.

Finally, we use `<div>` as a last resort, when there is no other appropriate markup tag.

14) `<section>` is a semantic markup tag, not a stylistic one

Expanding on what we said previously about the `<section>` element, it is important to emphasize that `<section>` is a semantic tag. In fact, it should have a heading tag, and even if it hasn't, using a heading would make sense.
It should not be used to tag a wrapper, a container or any other purely stylistic block.
So, for example, this is not a correct use of the `<section>` tag:

```
<section id="wrapper">
  <section class="container-fluid">
    <div id="main">
    </div>
  </section>
</section>
```

This is better, but it overuses the `<div>` tag:

```
<div id="wrapper">
  <div class="container-fluid">
    <div id="main">
    </div>
  </div>
</div>
```

An even better approach:

```
<body id="wrapper">
  <div class="container-fluid">
    <main id="main">
    </main>
  </div>
</body>
```

15 – The `<figure>` element

The figure element is mostly used with pictures, however it has a wider range of possible uses. Anything related with the document's topic, but it could be positioned anywhere in the document, could be wrapped in a `<figure>` element.

Think of illustrations, tables or diagrams in a book.

An interesting characteristic of `<figure>` is that it does not contribute to the document's outline
So it can be used to group elements with a common theme, such as a number of images with one common `<figcaption>`, or even a block of code.

16 – Grouping elements with `<figure>`: Use of `<figcaption>`

The `<figcaption>` caption should go either directly after the opening `<figure>` tag, or directly before the closing `</figure>` tag.

```
<figure>
  <img src="img1.jpg" alt="First image">
  <img src="img2.jpg" alt="Second image">
  <img src="img3.jpg" alt="Third image">
  <img src="img4.jpg" alt="Fourth image">
  <figcaption>Four images related to a topic </figcaption>
</figure>
```

D. Use of the appropriate tag and/or attribute for your intended purpose

17 – Tags that denote style are deprecated in HTML5 – use CSS instead.

Do not use `<big>`, `<center>`, `<strike>`, `<blink>`, cause they are deprecated – you should never use `<blink>` even if it was not deprecated!

Do not use `<hgroup>`, it is obsolete.
Do not use `<i>` for text in italics, `<b>` for bold and `<em>` for emphasis: the purpose of these elements has been redefined in HTML5

In general, as a best practice, it would be optimal to avoid stylistic elements in markup; that's what CSS is for.

18 – The `<br>` element is not for layout

Do not use the `<br>` to format your document or to add space between elements.
A rule of thumb here would be that, if it can be formatted by defining margin or padding in CSS, then you should not use `<br>`. If, however, you want to add line breaks within the same element, then `<br>` is appropriate:

```
<label>Please use the following text area:<br>
  <textarea name="loremipsum"></textarea>
</label>
```

19 – Type attribute is not necessary for stylesheets and JavaScript scripts.

In HTML5, there is no need to define the type for `<style>` and `<script>` elements. All modern browsers expect that stylesheets will be CSS and scripts will be JavaScript. It is still a very common practice, since many popular CMS add these attributes automatically, but there's no reason to do it in manually written code

Consider this:

```
<link rel="stylesheet" href="style.css" />
```

```
<script src="script.js"></script>
```

Instead of this:

```
<link type="text/css" rel="stylesheet" href="css/styles.css" />
<script type="text/javascript" src="js/scripts.js"></script>
```

20 – Use alt attribute for your images

It is good practice to always use an “alt” attribute for your images. It provides an alternate text for when the image loading is disabled on the browser and it is extensively used by screen-readers.

21 – Be careful when you use a “title” attribute.

The “title” attribute is not interchangeable with the “alt” attribute. Alt is used instead of the image, while title is shown together with the image, usually as a tooltip.

The HTML5.1 recommendation warns against overusing the “title” attribute, due to lack of compatibility with a big percentage of browsers, like touch-only browsers in tablets and phones:

This is an adequate use of the title attribute:

```
<input type="text" title="search">
<input type="submit" value="search">
```

The following uses should be avoided:

```
<a href="text.txt" title="Relevant document">txt</a>
<img src="img.jpg" title="My photo" />
```

Instead consider appropriately naming your link and using alt attribute for your picture:

```
<a href="text.txt">Relevant document</a>
```

```
<img src="img.jpg" alt="My photo" />
```

E. Code readability and formatting choices.

Everything mentioned in this category falls into the field of subjective. Each front-end developer ends up with their personal style, and this is fine as long as the style is consistent and makes sense. Generally, if your code is clear and readable by another developer, then this is good enough.

However, we include some general recommendations that seem to be widely accepted:

22 – Indentation consistency.

A code with either complete lack of indentation, or inconsistent indentation lacks in readability. For reference Google Style Guide for HTML recommends using two spaces for indentation and to not use tabs

If this was the widely accepted standard, then everyone would format their HTML code consistently. However, every developer is more likely to do things their way, so at least we should all try and keep it consistent: if you decided to use 4 spaces, or two tabs for indentation (please don't use two tabs), enforce it to every single line of HTML code you write.

23 – lowercase, Title Case, CamelCase or ALL CAPS?

- CamelCase is used in JavaScript and it is a visually identifiable JS formatting. For that, it is best to not use it for any snippet that is not in JS
- Title Case is only for text, strings, content. While it is not technically wrong to name your classes or IDs with mixed cased names, it does affect readability.
- ALL CAPS: again, nothing technically wrong with it, other than being conventionally considered "shouting", or visually unpleasant
- lowercase is the most widely used convention

24 – Quotation marks

The single quote or double quote argument. There are many arguments for both sides, however it really boils down to your personal preference. Whatever you decide to use, keep it consistent.

One note, though: That HTML allows single quotes, becomes very handy when generating HTML output from PHP. However, in manually written HTML, it is really question of preference, as long as it is consistent.

25 – `<pre>` and `<code>`

It is a usual practice `<code>` element with `<pre>` tags. The `<code>` element in itself, simply denotes computer code and does nothing to preserve the code formatting, while the `<pre>` element (preformatted text), will preserve any newlines and white spaces.

One thing to keep in mind is that, the `<pre>` element will preserve every newline. So, this code:

```
<pre><code>
.container {
  margin: 0 auto;
}
</code></pre>
```

will format your code block starting with an empty line. Use `<pre>` elements like this instead:

```
<pre>&lt;nav class="main_nav"&gt;
</pre>
```

26 – Character entities

Use character entities with `<pre>` when displaying HTML code.
Use `&lt;` and `&gt;` instead of < and > and `&quot;` instead of "
Outside of preformatted text blocks, the UTF-8 character encoding allows almost all characters (including ©, ® and even emoticons) to be typed in directly. However, it is a usual practice to escape `&`, <, >, " and ' even

outside preformatted blocks.

27 – Dash or underscore for class and ID names?

You can never use spaces in the id attribute or a class name

There is no other restriction, so the dash or underscore debate lies, again, on personal preference. One thing to note about this is that this personal preference might go in hand with one's preferred text editor. In Notepad++ a word with a dash character is considered one word and will be selected as one word. In Vim, dash breaks the word, while underscore does not break the word. So, when deciding which naming convention you will use, take into account the behavior of your favorite text editor.

28 – Comments

Comments might affect code readability in a positive manner, when used correctly. I have the habit of commenting closing tags (especially `<div>` closing tags) noting the class name of the opening tag – this makes it easy to know which block has been closed in nested tags.
Example:

```
<div class="myclass">
  <div class="nextclass">
  ...
  </div><!-- .nextclass" -->
</div><!-- .myclass -->
```

F) Validating and minifying.

29 – Validating

W3C validator sets the industry standards, so it should be the first place to check whether your code is valid.
W3C validator

You can also get the source code of their Nu validator:
Source code

W3C also offers a mobile-ready checker, that is still on alpha stage: [Mobile-ready checker]

30 – Minifying and combining CSS and JS files

A modern website will usually have more than one CSS files. The main stylesheet, a bootstrap or other grid stylesheet, maybe a few plugins or themes stylesheets, etc. Each CSS file makes a separate HTTP request, slowing down the load time of your site.

It is a recommended practice, in the final product, to minify and combine all your CSS files for improved load times. It is also usual to keep the unminified file, possibly in a "css/src" folder, because editing/debugging minified files is difficult.

Similarly, it is recommended to minify and combine your JavaScript files. It is also a recommended practice to move them to the bottom of the document, just above the closing `</body>` tag, so that the browser starts to load them after it has served the rest of the document.

Conclusion

I hope you like our html5 coding guidelines and front end web development best practices. If you find any errors or mistakes then do let us know. Thanks for reading

Sources:

- hail2u recommendations
- w3.org
- Dive into HTML5 from html5doctor
- html5doctor
- Google's style guide
- Various questions on Stack Overflow