50 HTML Best Practices & Guidelines to Build Better Web Projects

Elson Correia Follow

Feb 11 · 14 min read ★

HTML is the backbone of any web application and despite being very easy to pick up, there a large number of tags containing easter eggs. Since HTML will not complain when

you do things wrong you may be affected by not following a certain good practice for your application.

Check Other Best Practices List

- **50 CSS Best Practices**
- **50 Javascript Best Practices**

1 — Avoid Inline Style

The only thing that can override an inline style is the CSS !important flag which is not a good practice (check 50 CSS best practices article for more details). The inline style should serve a very specific purpose like some style manipulation done from the Javascript side but not as a way to style your HTML.

2 — Add critical style first the rest later

If you put all the styles of the site in one file it can take a long time to fetch and parse which can delay your site render. What you should do is include the main and basic styles for the site in a style tag in the head or a smaller stylesheet to load first — **just the needed style for the first render** — then an external stylesheet with the secondary styles but **deferred**.

Secondary styles can be any style for things that require user interaction to be revealed like modals, dropdown, and notification components. Even things far down the page that a user needs to scroll to.

3 — Avoid Script tag with code when possible

A lot of SDKs will show you an example of script code to include in the header of your site but you can get away with just putting them in an external script tag sometimes and control how it loads. If you need to use something from these just make sure it loads before you get to use them but in general, you should avoid mixing Javascript code with your HTML.

4 — Script Tags go at the bottom

This rule is even better for the **browser that does not support certain optimization attributes** like defer and async. In general, it is a good practice to include your script tags last to give the browser time to finish parsing and rendering HTML and CSS if they are not async or defered.

If you are generating HTML and CSS from Javascript — like with React, Angular, and Vue — you definitely should put the script tag last.

5 — Reduce the number of external links

Always try to combine your external stylesheets and script files into a minified file. You can always defer or preload/prefetch. This goes to any file links in the header as well as script files. Have a few of them and always tell the browser how to handle them.

```
// tell the browser which links to prioritize
<link rel="preload" ... />
<link rel="prefetch" ... />

// tell the browser to fetch while parsing the page
<script async />

// tell the browser you need this after parsing
<script defer/>
```

6 — Always add an alt tag to images

The alt tag has the alternative text to show in case the image fails to load. It also provides the extra context of what the image is and it is great for your SEO. It makes your images searchable and identifiable.

7 — Don't forget the Page Title

Not just the page title but many other titles of many HTML tags. Page titles are used in Search Results on search engine pages like Google and Bing. The title is used in the tabs for people to see and it is used by many social media sites when people paste your site as a way to share your site.

Make the title describe the content of the page in few words. It may contain the name of the site, the title of an article, or a simple title of the page.

```
<title>Before Semicolon – About Page</title>
<title>Before Semicolon – Tech Videos & Blog</title>
<title>My Super Cool Article Tittle</title>
```

8 — Some elements tags require a title

You can title many HTML tags which help them even further, for example:

- You can specify alternate stylesheets and by adding title attribute you let the user with a nice description of what the style is for.

```
<link href="reset.css" rel="stylesheet" type="text/css">
<link href="default.css" rel="stylesheet" title="Default Style">

Alternate stylesheet
<link href="fancy.css" rel="alternate stylesheet" title="Fancy">
<link href="basic.css" rel="alternate stylesheet" title="Basic">
```

- If you use the <abbr> tag don't forget the title because it is used as a tooltip for people to find out what the abbreviation is for.

```
<abbr title="Cascading Style Sheets">CSS</abbr>
```

- An input field can be enhanced with a title that is used as a tooltip to explain the field as well. A title attribute and a label for the input serve different purposes. **A good use case is to include a title for input with the pattern attribute specified.**

```
<input type="password" id="pwd" name="pwd"
  pattern=".{8,}" title="Eight or more characters">
```

9 — One h1 tag per page?

If your page includes the logo of the website you may want to use the h1 tag to wrap the logo site name as the top main heading. You can also use multiple h1 tags to highlight multiple top points of the page. **This rule is non-critical for your website** and it will *depend on your preference as long as you don't wrongly nest heading tags.*

10 — Include meta tags

Meta tags can help your site a lot. Everyone who builds websites with SEO in mind knows how important they can be. Meta tags are not just for SEO but to communicate with your browser about certain features you want. Some social media accounts require some type of metatags.

```
// tells the browser about the charset you want to use
<meta charset="utf-8">

// Tells the browser to redirect to
// beforesemicolon.com after 3 seconds
<meta http-equiv="refresh"
content="3;url=https://beforesemicolon.com">
```

```
// tells twitter about your twitter acount
// when someone share your site link
<meta name="twitter:site" content="@beforesemicolon">

// tells the browser your site is responsive
<meta name="viewport" content="width=device-width, initial-scale=1">
```

11 — Add prefetch font links to the head

The best way to load your site fonts is by including them in the head with the link tag and the prefetch/preload attribute. You can still include the CSS @font-face rule in your external stylesheet.

12 — Always use responsive meta tags

You should always make your site responsive. Now people can request a desktop or mobile experience through the browser and the minimum you can do is make your site fluid but, responsiveness is a must.

```
<meta name="viewport" content="width=device-width, initial-scale=1">
```

13 — Custom component needs to be accessible

Whenever you want to build custom components to replace built-in elements like select tag, button, range input knows that you are throwing away accessibility features by doing so. If it is custom, add accessibility tags and follow accessibility guidelines.

Learn how to build various custom elements with accessibility

14 — Prefer default role and built-in accessibility

You don't need to include certain roles to a lot of HTML5 tags and you should customize/style HTML5 elements as much as possible instead of creating alternatives.

```
// these are semantic tags
// they don't need role
<nav role="navigation">...</nav>
<button role="button">...</button>

// you may need to do this if your link
// works like a button
<a href="" role="button"/>
```

15 — Avoid adding HTML tags for styling purposes

HTML structures your site and you should avoid extra markup for styling purposes. CSS is super powerful and allows incredible looks without changing the content. That is what the CSS Zen Garden was trying to prove. HTML is to markup content, not to make the style your content.

There may be times you need to break this rule but in general, changing your HTML should not be the number one popping in your head when trying to style things.

16 — Avoid formatting the text yourself

Add content in the format it needs to be. Don't try to tweak content to achieve certain formats. There are many tags and CSS properties that allow you to change how the text should look like.

17 — Use appropriate containers tags (div vs section vs article)

You will benefit a lot from understanding **when to use Section vs Article vs Div in Html**. Understanding when to use these tags has a great impact on your site and you should avoid using div tags everywhere. HTML5 is full of semantic tags.

18 — Always go for semantic tags

If there is a semantic tag you can use, prefer it. Always have your HTML that can speak for itself which helps you avoid unnecessary roles, and classes. Some of these tags have some minimal style that can help your content and write even less CSS.

19 — Dont use the IMG tag for non-content imagery

You should learn when to use HTML image tag vs CSS background image as soon as possible. A common mistake you may find out there is developers using the image tag to include non-content imagery like icons. If the image does not help the content make sense it should not be an image tag and a simple test is by looking at the content without the image. If it makes sense then the image is decorative.

20 — Always close or self-close the tags

The browser tries its best to close tags for you but it does not always get it right. Some tags are required to be closed(html, head, body, p, dd, li, option, etc) and others are forbidden to be closed (img, input, br, hr, meta, etc).

21 — Add favicon file to the root directory

The browser will automatically fetch a favicon for you. **You don't even need to specify the link for the favicon**, just include it in the root of your site. Whatever you do, always include the favicon for your site explicitly and of many sizes and target.

When the tabs are too small to read the title, the favicon becomes useful. You should try to use a favicon generator to support many other devices, browsers, and other situations they are used.

22 — Add a manifest

Including a manifest is one of the required steps to create PWA and I think that in general, you should always include it because it further informs the browser about how to handle your site.

You can specify short and long names for the site, how it should be rendered and all the icon sizes to use in tabs, and when people bookmark or add your site to their mobile devices' home screen. Try one of many manifest generators out there.

23 —Always specify the DOCTYPE

It may seem unnecessary but including the HTML attribute in the DOCTYPE will ensure the browser render your stuff correctly. It tells the browser you designed the site for modern browsers and it should not bother with quirks mode.

```
<!DOCTYPE html>
```

24 — Page language

By specifying the language of your site will help screen readers pick the right language to announce. Browsers also use it to determine if they should auto-translate your site or not. The lang attribute should describe the language used by the majority of the content of the site.

```
<html lang="en">
```

25 — Escape special chars (‘&<>”)

HTML has some specific characters that mean something to it like the less and greater symbols. By not doing so you may break how your site renders especially if you are using content coming from untrusted sources or user input.

26 — Use data-* correctly

Don’t use the data attribute to include sensitive data or things that can class and other attributes can be more appropriate for. You can learn about them in this awesome CSS

Tricks article.

27 — Add DateTime to time tag

An easy thing to forget is the datetime attribute on the time tag. This small detail helps your devices find dates in the content that you can easily click to add to your calendar for example.

```
<time datetime="20:00">20:00</time>
```

28 — You can omit the “for” and “placeholder” attribute when used together

You don’t need the for and the placeholder attributes if you wrap your fields inside a label tag. The nesting makes them implicitly bound together and It is a great way to style inputs as well, you can check an example of how to use this in the **input style technique video** for more details.

```
<label>
  First Name:
  <input type="text"/>
</label>
```

29 — Stop using <hgroup/> tag

HTML5 gave us cool tags and some of them ended up going away. The hgroup tag can seem cool but you should stop or never use it. There are many others you should not try because they got deprecated. Here is a full list of tags which include some that you probably will never use.

30 — Follow a consistent HTML format

It is important to remain consistent in your HTML style. You can use prettier to help you with that but the goal is to always follow a consistent way you code your markup.

31 — Avoid SIBU tags

The SIBU tags are *<s/>*, *<i/>*, *<b/>* and *<u/>* tags which are for style purposes only, they are not semantic tags at all and anything they do can be accomplished with CSS alone or other semantic tags.

- ** instead of *<b/>*
- ** instead of *<i/>*

- **text-decoration: underline** instead of <u/>
- **text-decoration: line-through** instead of <s/>

```
// Don't
<i class="icon-name"></i>
<p>A simple equation: <b>x</b> = <b>y</b> + 2 </p>

// Do
<span class="icon-name" aria-hidden="true"></span>
<p>A simple equation: <var>x</var> = <var>y</var> + 2 </p>
```

32 — Omit boolean attribute values

Some attributes don't need value so you should not need to include them. They are called boolean attributes and their values are the same as the attribute name so you can go ahead and omit it.

```
<button type="button" disabled/>
<input type='checkbox' checked/>
```

33 — You can put social links inside the address tag

The address tag should be used to *wrap any information related to how people should reach or get in touch with you* — mailing address, phone numbers, email, website, social media links (if contextual).

34 — Dont put block tags inside inline tags

Think of block tags as boxes and inline tags as envelopes. Don't try to put boxes inside envelopes. Some browsers in some situations will remove the block tag from inside the inline tag breaking your markup. Block and inline tags are not to be confused with CSS display block and inline. In the eyes of HTML tags will always be inline or block regardless of their CSS display value.

35 — Beware the base meta tag

The base tag is really misunderstood and before you decide to use it I recommend you learn about it because it is easy to mess up links on your site and it is not intuitive what is going on.

It can be super handy and equally create problems if misused.

```
<base href="https://www.beforesemicolon.com/">
<base href="/">
```

36—Continue to use tables

Definitely dont use tables for layout (email layout devs still use it) and if you have to use tables for layout add the role of presentation. The table is the perfect choice to represent tabular data and its accessibility is of tabular content so, continue to use the table. If you want table behavior but dont want to use table tags, use the CSS display property which has options for the table.

37 — Get in the habit of using role and aria attributes

You should become familiar with role and aria attributes and use them whenever possible. They are useless for some HTML elements but add them to any custom thing you create in order to make your custom components accessible for any user and screen reading technology.

Learn how to build accessible components

38—Comments?

Comments can be really useful but why comment HTML right? Use comments at the end of a container to make it easier to spot things when they are collapsed. Also, comment message for you later and never ship commented out markup.

```
<main>...
  <section> ...
  </section><!-- about section -->
</main><!-- main section -->
```

39 — Always specify multiple backup sources for media

For video and audio always include multiple sources.

```
<audio controls>
  <source src="myAudio.mp3" type="audio/mpeg">
  <source src="myAudio.ogg" type="audio/ogg">
  <p>Your browser doesn't support HTML5 audio.</p>
</audio>

<video controls width="250">
  <source src="flower.webm" type="video/webm">
  <source src="flower.mp4" type="video/mp4">
```

```
  <p>Your browser doesn't support HTML5 video.</p>
</video>
```

40 — Prefer picture over images where it applies

The picture tag allows you to specify multiple resources for different viewport sizes which allows you to load appropriate images faster and use different images for the appropriate view sizes. This allows you to load smaller size images for mobile devices faster and bigger images for desktop.

```
<picture>
    <source srcset="surfer-240-200.jpg" media="(min-width: 800px)">
    <img src="painted-hand-298-332.jpg" alt=""/>
</picture>
```

41— Stop supporting IE

Whatever comments or tags you add to your HTML in order to support IE, just stop! Unless you strictly have to build for that browser, try to recommend other browsers to that user instead of adding some hacky thing to your markup for IE. Microsoft is stopping support for IE11 in mid-2021.

42— Avoid too much HTML

Too much HTML is not good. Adopt strategies that allow you to load just enough HTML for initial render and have the rest on a different page or that can be later fetched on a scroll or some button click by Javascript. Too much HTML means longer parsing time and just unnecessary in general.

43—Lazy load images

Some browsers will load the images only if they are in the view that way if you have a page with 100 images, only those within the viewport will be loaded, and as the user scrolls the rest gets loaded accordingly. All you gotta do is specify the loading attribute with a value of lazy. There is also a polyfill if you want this feature in all browsers.

```
<img src="image.jpg" alt="..." loading="lazy">
```

44 — Always specify preload option for video

You can also lazy load videos on the page with the preload attribute. It is best practice to always include this attribute because different browsers have different defaults. Preload

of none will prevent the browser from loading the video immediately and show the placeholder meanwhile.

```
<video controls preload="none" poster="simple-placeholder.jpg">
  <source src="one-does-not-simply.webm" type="video/webm">
  <source src="one-does-not-simply.mp4" type="video/mp4">
</video>
```

45 — Videos are better than gif at times

Often times developers use a gif instead of a video believing the gif can be lighter but depending on the video and format, videos can be a better option so, always compare your video and the gif of it before you make the decision.

46 — Always specify the button type

Simple rule! Always specify the type of the button. The button is of type “submit” by default which is not always the behavior you want. In general, always be explicit about the type of things you want, even it is the default value.

```
<button type="button">My Button</button>
<input type="text"/>
```

47 — Blockquote wraps the tag containing the text and the q tag is almost useless

You should not wrap the text with the blockquote tag directly. Put the text inside a p tag then wrap the p tag with the blockquote. The q tag will simply add quotes to the text. It is great for citation but in general people prefer anchor tags for citation. You can just add quotes yourself or use it to target quoted text in the CSS.

```
<blockquote cite="https://www.huxley.net/bnw/four.html">
  <p>Words can be like X-rays, if you use them properly—they’ll go
```

```
through anything. You read and you're pierced.</p>
</blockquote>

"some quoted text"
// same as
<q>some quoted text</q>
```

48— Avoid Div and Span tags

You can almost always find semantic tags to use instead of div and span tags. People normally use them to wrap certain content and I am guilty of it too, especially when I am using Javascript frameworks/libraries. In general, always prefer semantic tags. Div is for blocks as span is for inline tags.

49 — Minify your HTML

Something you don't often hear but it is not just your CSS and Javascript you need to minify. You need to minify it, especially if your site is static.

50 — Validate your HTML

Always validate your HTML to make sure it is valid markup. It prevents you from making dumb mistakes and with time you will learn enough to avoid these mistakes altogether. This can even be added to your build process to ensure your website does not get deployed with invalid markup.

Conclusion

Best practices rules are common rules followed by a community but every experience is unique. You should always test and experiment with different rules in order to find something that makes sense for you, your team, and the type of project you have in hand.

Youtube Channel: Before Semicolon
Website: beforesemicolon.com

HTML Web Development Technology Frontend Best Practices