



Web Technical Guidelines for Vendors v1.2

Brief

This document outlines some of the recommendations that we have when building web-based applications or sites for HubSpot. These guidelines are not intended to be an exhaustive list or comprehensive documentation for building projects, instead they serve as suggestions and advice for common pitfalls.

For acceptance criteria for projects, see the [requirements document](#).

Areas Covered

1. General Advice, HubSpot CMS
2. Legal Compliance and GDPR
3. Accessibility
4. Performance

General Advice, HubSpot CMS

- Don't target **HubSpot classes** in your project's CSS. Use your own custom classes - the default classes provided by HubSpot are subject to change outside the scope of your project
- Include a list of all **custom CTA's** that you may have created with your submission, as these cannot be copied across portals and have to be manually recreated.
- If using **custom CTA's**, don't target the custom CTA classes directly. When migrating your project into our CMS portal, we will need to recreate the CTA's, and their IDs will not match the classes you might have set up your CSS to target. Use custom your own CSS classes instead
- **Localisation** of content for different regions is often a requirement with these projects. It's best practice not to hardcode strings or use images that contain text, as these cannot be translated. In general, all strings, images, and text content should be editable via the CMS page editor, and not hard-coded into HTML, CSS or JS.

- Modules should be reusable - for example if you have a module that is repeated across multiple pages, that module's CSS, Javascript and markup should be reused, rather than multiple versions of the same module existing. This creates a more consistent user experience while reducing the amount of content we need to load on the page.
- This point also applies to CSS classes - for modules that look, feel, and behave the same, it's good practice to move the common CSS into a shared CSS file in the CMS and load it using the `require\_css` option. That way, the amount of CSS we need to load is greatly reduced and the code is much more [DRY](#).
 - For example, for an accordion module on a project home page, you might use a class like **.home-footer-accordion-wrapper**. But if this module appears on multiple pages, you might end up needing to classes to describe the same module on a different page: **.lp-mid-accordion-wrapper**. In this case, it would be better to make this class more generic, **.accordion-wrapper**, and move it to a shared CSS file that can be loaded on all pages where it is needed.
- Do not hardcode styles or scripts in the DOM or in the head/footer tags in the design manager. For global styles or styles that are not specific to a single modules, create a separate file and [import it using the CMS import feature](#)
- If working on a new version of a project that your team has worked on previously, make sure you separate old project files from the new or use a different naming convention that will distinguish one project from another. This prevents major issues with conflicting file names or loading unnecessary files when we try to import the project.

Legal Compliance and GDPR

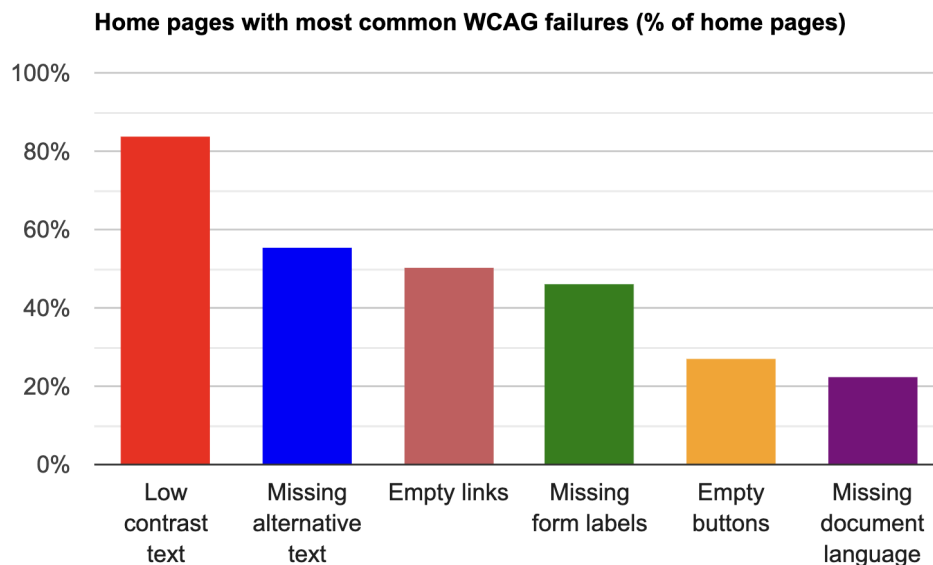
All of our web applications must comply with the European [General Data Protection Regulations](#) (GDPR).

GDPR applies to the processing and/or storage of any kind of personal data - for example; form submissions, personally identifying cookies or data in local-storage or the storage of personal information in databases like HubDB.

For any questions relating to legal compliance, please contact [Laura Moloney](#).

Accessibility

- It's vital that anything you build complies with the **WCAG 2.1 A,AA** standards. You are responsible for adhering to these standards during your development period. Please refer to the [requirements document](#) for more information on testing and tooling required for Accessibility acceptance criteria.
- The most common issues we see regarding accessibility are similar to those in the [WebAIM Million](#):



- These issues are readily testable with [AXE devtools](#) and [Google lighthouse](#).

- It's important that you write valid HTML. Valid HTML serves as the foundation for building accessible web applications. The [WC3 HTML Validator](#) is relatively straightforward to use and validation errors are generally easy to remedy, you should use this to verify any HTML that you write.
- Also important regarding markup is that your HTML is semantic. Though semantics can be somewhat subjective, it's important that we enable browsers and search engines to better interpret our content. [Read more about semantic html and accessibility here](#)

Performance

- Make sure you read [the HubSpot documentation on optimizing CMS sites for performance](#). These cover all aspects of leveraging the CMS features for performance
- [Google Lighthouse](#) should be your testing platform for performance. It will flag issues, and provide details about how to resolve them.
- Do not use images that have not been optimized for the web to reduce their file size. Below the fold images should always be lazily loaded using the native **loading="lazy"** HTML property. For more information on lazy loading, [see the MDN documentation](#)