Developer-friendly WCAG checklist

Accessible by Design

Checking your web pages against the below-listed recommendations before sending for manual accessibility testing, will significantly reduce the accessibility errors.

I compiled common errors and provided explanation for each of them.

Armagan Tekdoner

IAAP CERTIFIED WAS

Web Accessibility Specialist
International Association *of* Accessibility Professionals

Web coding showcase: grifare.net

All functionality should be operable using keyboard only

This is probably the most important checkpoint. Failure to comply completely stops users with physical disabilities.

See if you can do everything using the keyboard (Using the tab, shift + tab, enter, space bar and arrows, never using a mouse).

One way to test keyboard accessibility is trying to complete a task. Here are some ideas and common failures:

1. If there is registration, can you register using the keyboard only?
2. If there are modal windows, alerts, pop-ups, dialogues that require user interaction, can you access them with keyboard?
3. Can you tab through the page and get back to the address bar?
4. Can you navigate inside menu / side menu items?

Semantic HTML5 elements

Try **using semantic HTML5 elements** instead of DIVs, this will prevent many accessibility errors without additional mark-up, such as ARIA. No tabindex attributes will be required as well.

Make sure the website passes the **mark-up validation**

When the page validates, in addition to satisfying the parsing criterion which has limited requirements, many other issues will have been fixed automatically. Using the mark-up validation tool, we almost always find parsing errors that should have been found and fixed by the developers and sometimes these are the root causes of multiple failures.

Avoid layout tables

Tables are intended for data presentation, the design should be handled by CSS.

Although using tables for layout is not an automatic accessibility error, it is difficult to achieve error-free results when tables are used.

&nbsp; and <br />

Usage of multiple whitespaces or more than one line break one after the other (<br />) for positioning elements, is a solid accessibility error. Use CSS instead.

Careful with the forms

Each and every input field must have its own associated label. Just properly using label and input elements with the correct attributes takes care of the vast majority of issues.

All screens must have unique titles

The author of the page should address this.

Page titles are important because they help a user navigate their browser. Most users have multiple tabs open at once. It is easier to jump between pages if each page title is unique.

The title should describe the web site as well as the specific page being displayed by the site.

The optimal format suggested is as follows:

```
Primary Keyword - Secondary Keyword | Brand Name
```

```
8-foot Green Widgets - Widgets & Tools | Widget World
```

Google typically displays the first 50–60 characters of a title tag. Mozilla has extensive coverage about titles.

Using automated tools

Use automated tools such as Axe or WAVE, check whether there are any **colour contrast failures**. Detecting contrast errors is one of the jobs the automated tools can perform successfully.

Responsive design

It is the key to ensuring the entire content is properly displayed on mobile devices and on smaller screens (by changing the viewport size).

Headings must be meaningful

And they should be used to create a hierarchical structure. (Heading levels should not be skipped.)