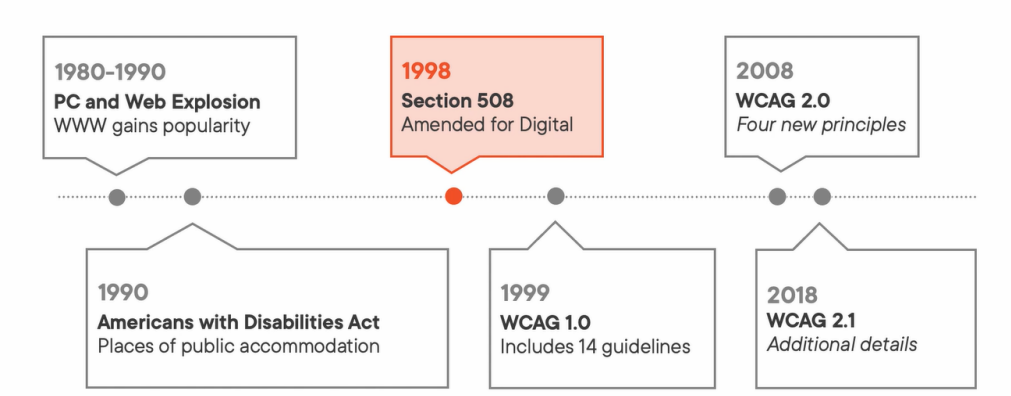
Accessibility

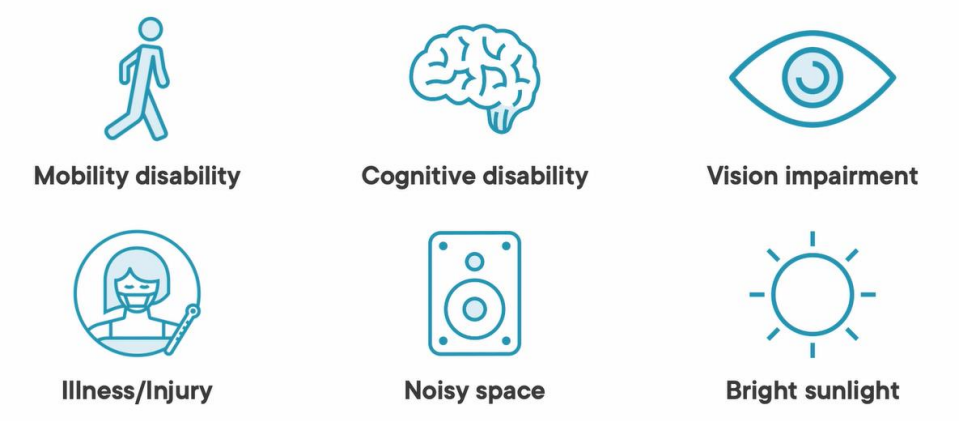
UX Accessibility

History of accessibility

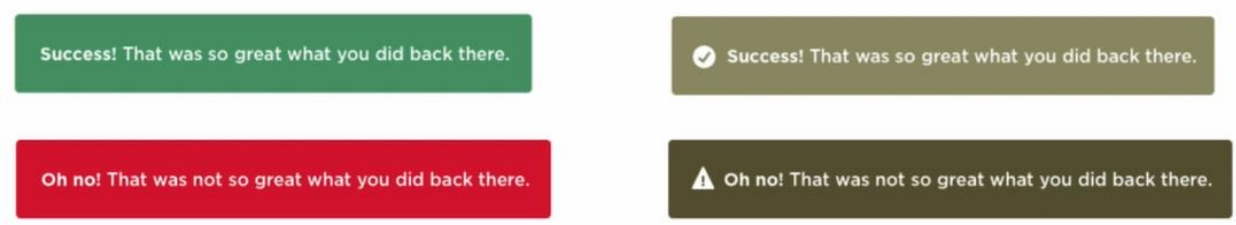


Your company can be sued for having poor accessibility on your website. No company is exempt – ADA

Who needs accessibility –



Don't use color alone to convey information. Means in below also use icon –



We should follow a proper content hierarchy.

Create accessible forms – each control should have label or aria-labelled-by, use grouping of controls, provide clear notifications, break up long forms. We should avoid placeholder text in your forms. It is often low contrast and difficult to see.

Assistive technology – screen readers, screen magnification software, speech input software, head pointers, eye tracking, single switch entry devices.

Architecting for Accessibility

Diversity of abilities – impairment (situational, temporary, permanent), disability and handicap.

Web accessibility guidelines – WCAG (A, AA, AAA).

JAWS is most commonly use screen reader for chrome browser then NVDA follows for Firefox.

When you incent, people optimize for reward. When you inspire, people optimize for purpose.

Think Differently About Accessibility

Neurodiversity – a range of different kinds of the brains. We should have this in our workplace.

Accessibility: Testing and Screen Reader Use

4 types of disability – physical, vision-related, cognitive and hearing-related.

Focusable elements – links, buttons, form elements – text input, text area, select, checkboxes, radio buttons.

On the website load, the first tab should go to the ‘skip link’, it generally skips the site navigation or repetitive contents.

The focus should be clearly visible by having proper contrast values.

If focused element is not visible then on the console we can type document.activeElement, it will give us the focused element.

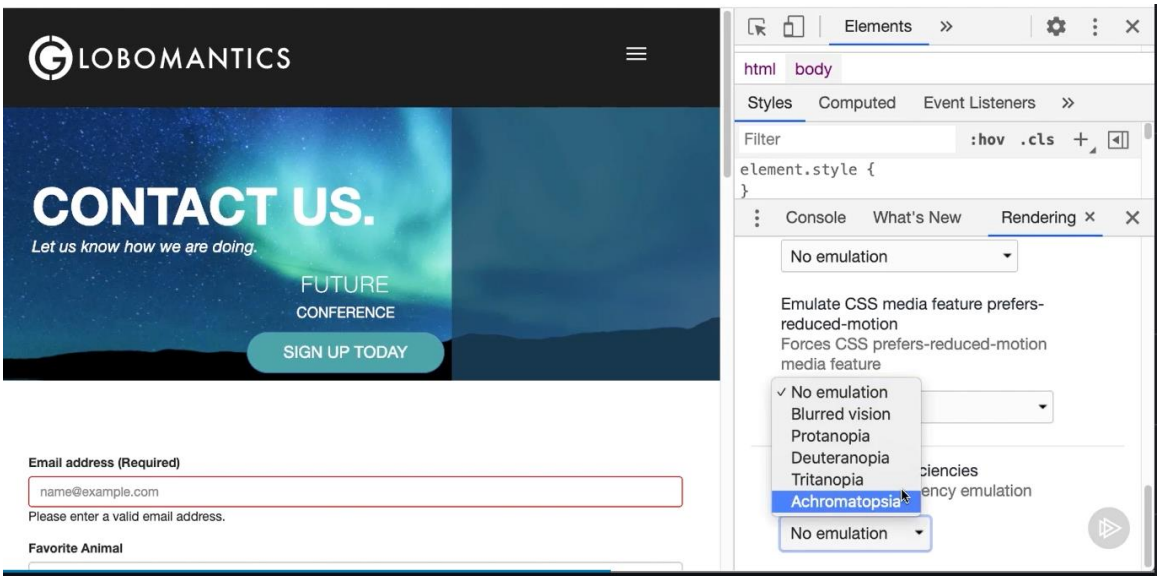
In radio button, if a radio is selected then we need to use the arrows to move around other options, but if none is selected then we can use tab key to move around other options. It is called widget navigation where too many focusable elements on a control, it provides skip links functionality.

Interactive form controls are activated with SPACE key. For links to activate use enter key.

For low vision user, the content should be zoomable, it should not lead to any loss of functionality and content.

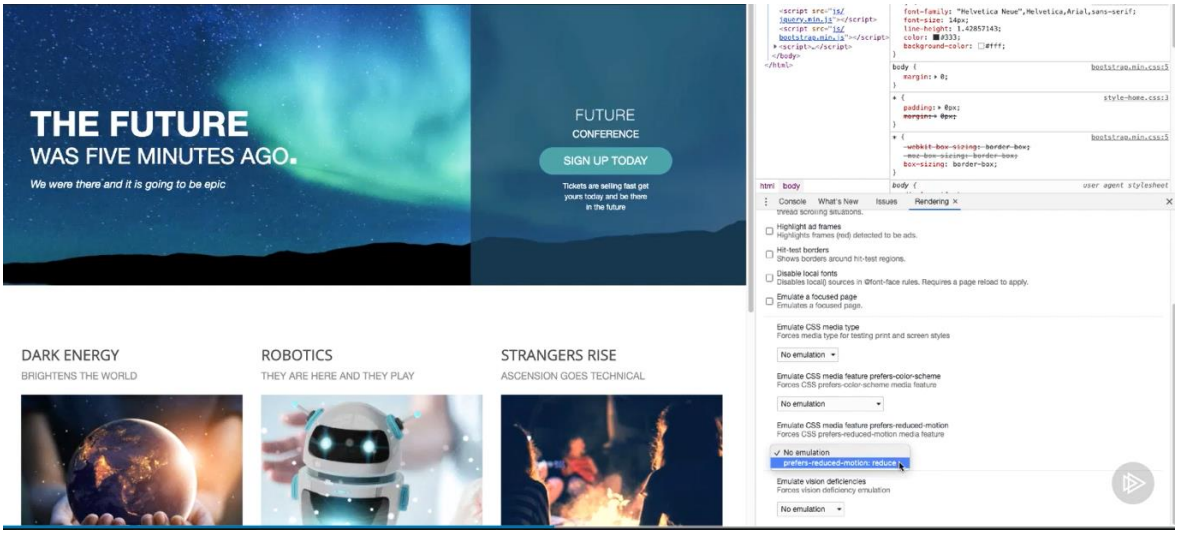
We can also increase just the text of the browser page instead of complete page content.

By going into the rendering tab, we can emulate different vision related tests –



The content on mouse hover or keyboard focus should be dismissal either from the original event from which it appeared or by using the ESC key. We should also be able to copy and paste the content from the tooltip.

Too much use of the animation can cause distractions among users, the website should also provide pause, stop and hide the animation. We can use chrome emulation for testing –



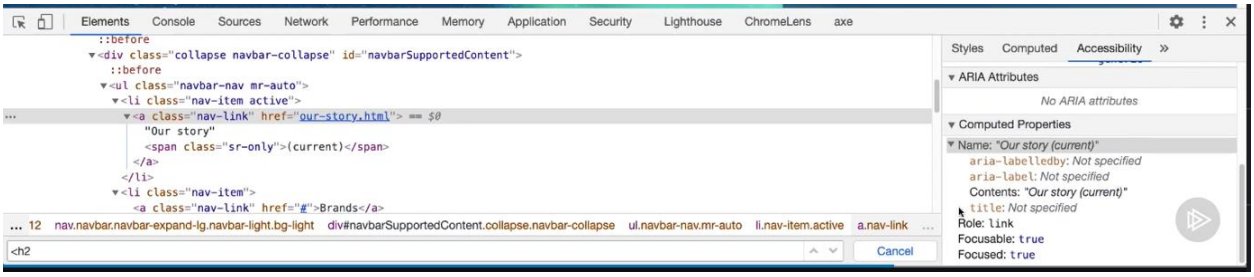
Video and audio should have transcripts, captions, audio description, pause/stop and hide buttons, avoid auto play.

We should have language, charset=utf-8, title tags on the HTML page. The title bar of browser should also include company name.

HTML landmarks such as header, nav, main, footer, aside, form and section should have accessible name by providing aria-label attribute.

Links are for navigation or change of context, buttons are for action.

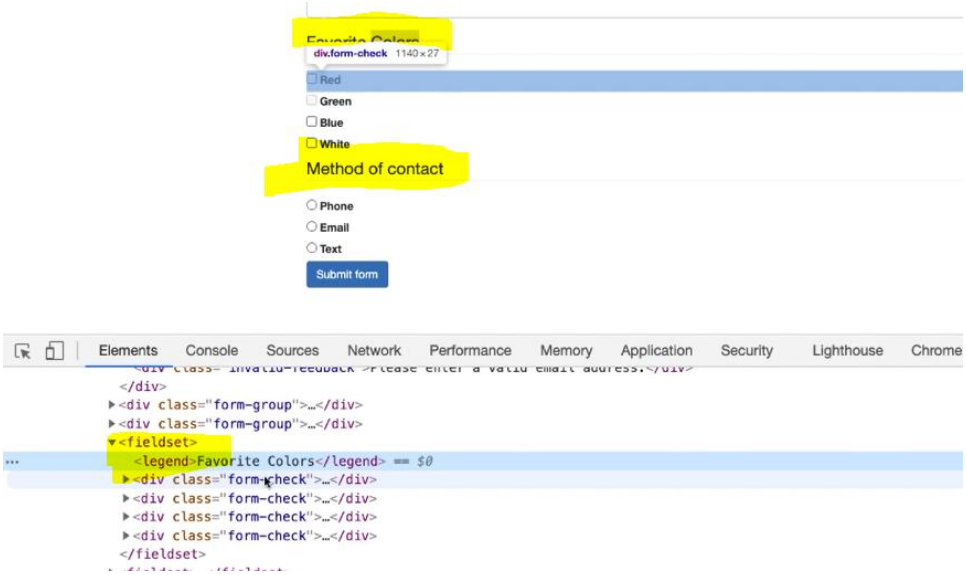
We can use accessibility tab in chrome dev tools to check computed property –



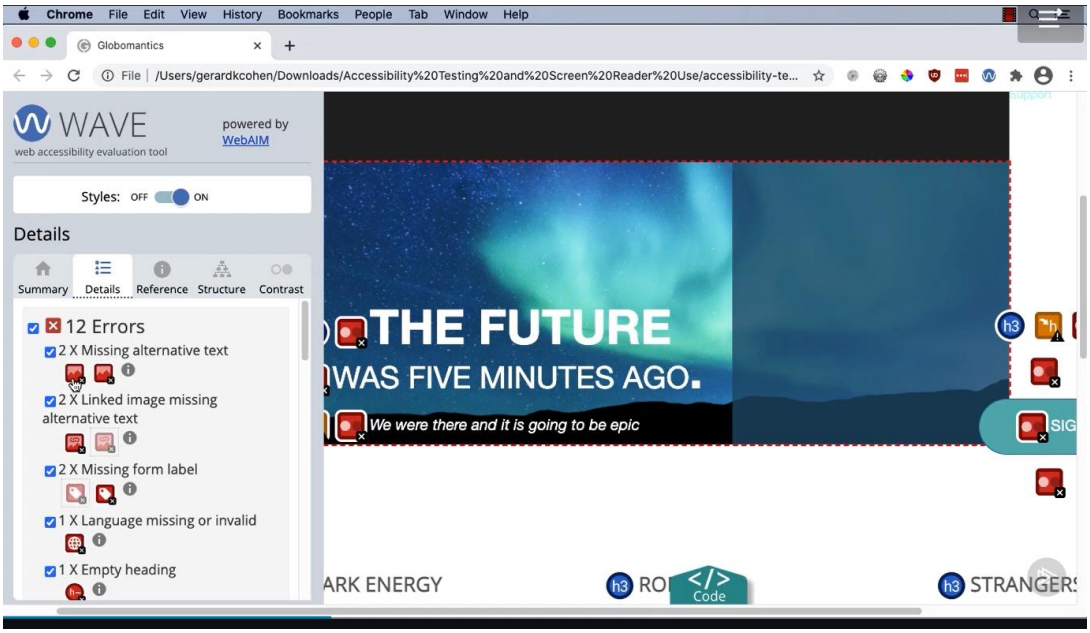
The table should have “scope” attribute as “col” or “row” as per the column or row direction. For accessibility we should avoid nested cells.

The email field should have autocomplete attribute as ‘email’.

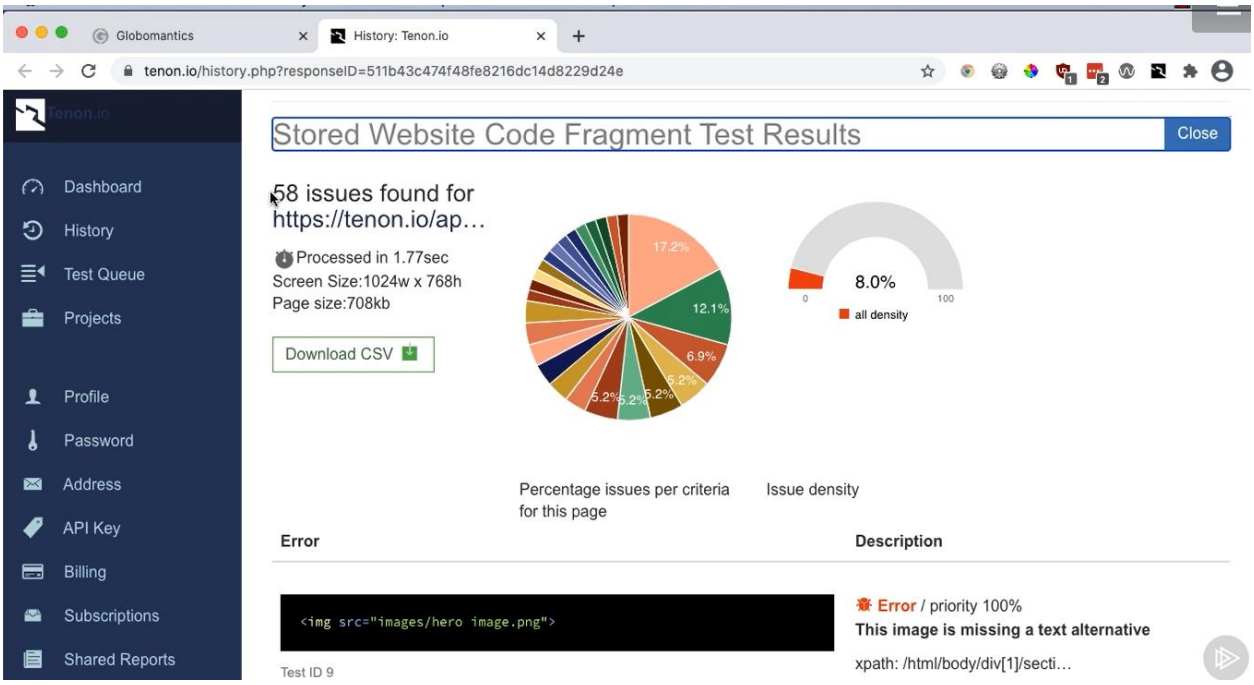
Group labels like for radio or checkboxes, should use <fieldset> and <legend> tags.



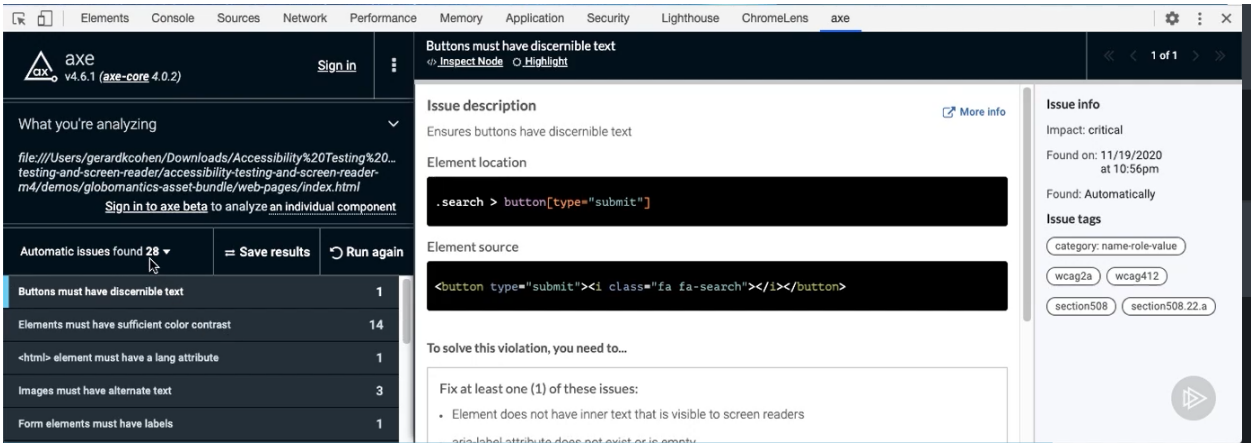
Testing tools - WAVE



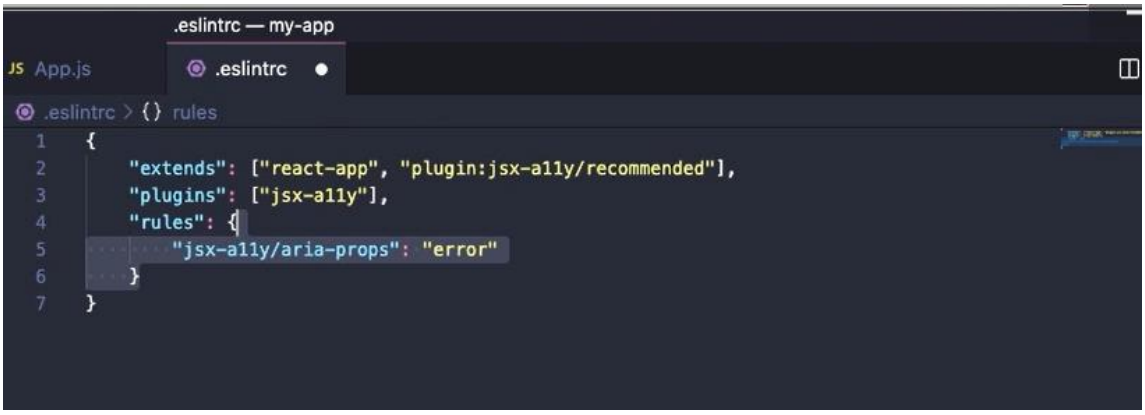
Tenon Check



Axe by Deque



Use eslint-plugin-jsx-ally to integrate the accessibility check into the build process.



We can also use axe-core plugin to integrate the accessibility test in our unit test cases frameworks.

Screen readers can also be used by low vision or cognitive users not just by blind ones.

Screen reader modes – browse mode,

Common single-key shortcuts –

Common Single-key Shortcuts

h
- Headings

1-6
- Heading levels

t
- Tables

b
- Buttons

g
- Graphics

TAB
- Interactive controls

SHIFT+ navigates in reverse

Navigating content keyboard shortcuts –

Navigating Content

UP/ DOWN
- Previous/ next line

MODIFIER + LEFT/ RIGHT
- Previous/ next word

LEFT/ RIGHT
- Previous/ next character

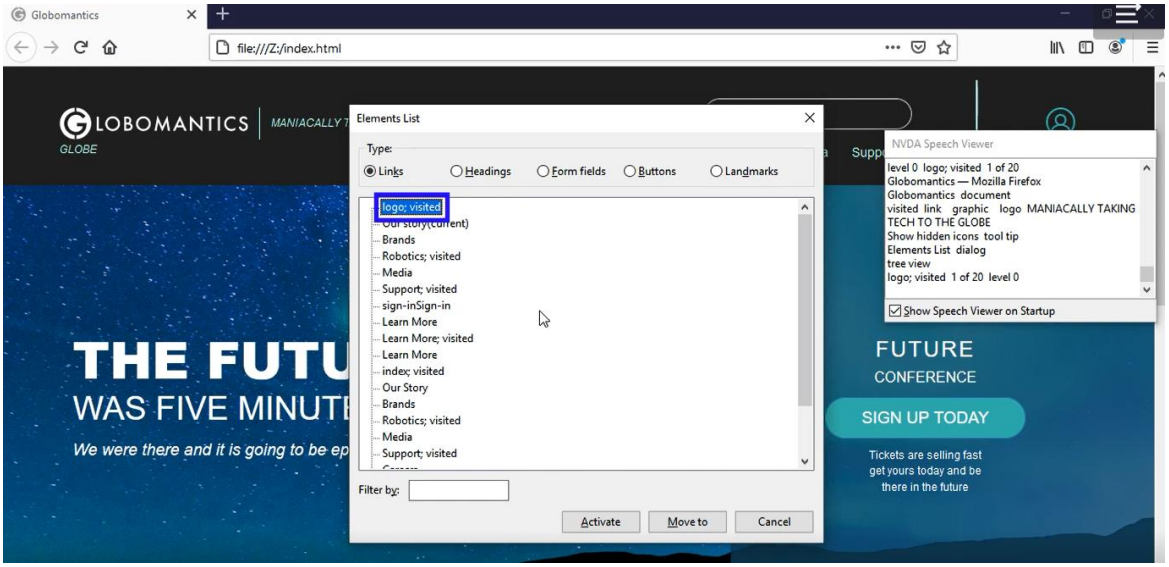
Forms/Focus mode – useful while interacting with form controls.

Application mode – combination of browser and forms mode.

Testing with NVDA – most preferred tool for window.

Press ‘ctrl’ key to make the NVDA turn off and turn on.

Using NVDA we can also use “Elements List” dialog –



JAWS - It is not free, but is another popular testing tool and most popular.

For macOS and iOS we can use VoiceOver tool.

Accessibility: Keyboard Input and Forms

The WCAG (web content accessibility guidelines) measure to accessibility of a website. Level A, Level AA, Level AAA. Who benefits – who only has one arm so can't operate mouse, only the keyboard, a person who is blind so uses a screen reader, a person who has dexterity problem, can't click on a small item, uses keyboard if she is struggling with the mouse.

Basic commands on form which can be interacted with like button, link and input controls, they are called focusable elements, pressing enter on that link or button should activate it. Space bar is used to toggle a checkbox and open a select control. Up and down arrows are used to scrolling the page or scroll through select component.

Mouse should be required only for drawing in an art program, some games, but drag and drop and resizing and rotating can be handled through the keyboard.

If we designing a complex interface then we can also add custom keyboard shortcuts to make it easier to use like below, it should be single key shortcuts -

Checking key combination use 2 key combination if it is going to critical function like deleting all record like above using ‘x’ is not an good choice which can be pressed accidently –

```
$(this).on('keydown', function (event) {
    if (event.key == 'r') {
        $(".members>div:focus").remove();
    } else if (event.key == 'x' && event.ctrlKey) {
        $(".members>div").remove();
    }
});
```

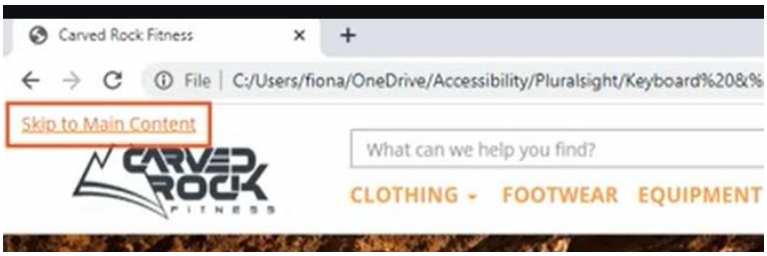
We can also change default focus outline border i.e. blue to something else like orange to match the theme of the website. Also to make the highlight of anchor link better, we can remove the box and replace it with underline.

If we set tab index as -1 then we cannot tab to it with keyboard, but can set focus programmatically. If we set 0 then we can tab to element and focus order determined by the HTML.

We can avoid keyboard trap like below, but we should avoid the key trap to occur at-all –

We can use the skip links to focus on main content, and make it hide and unhide if it gets focused by keyboard –

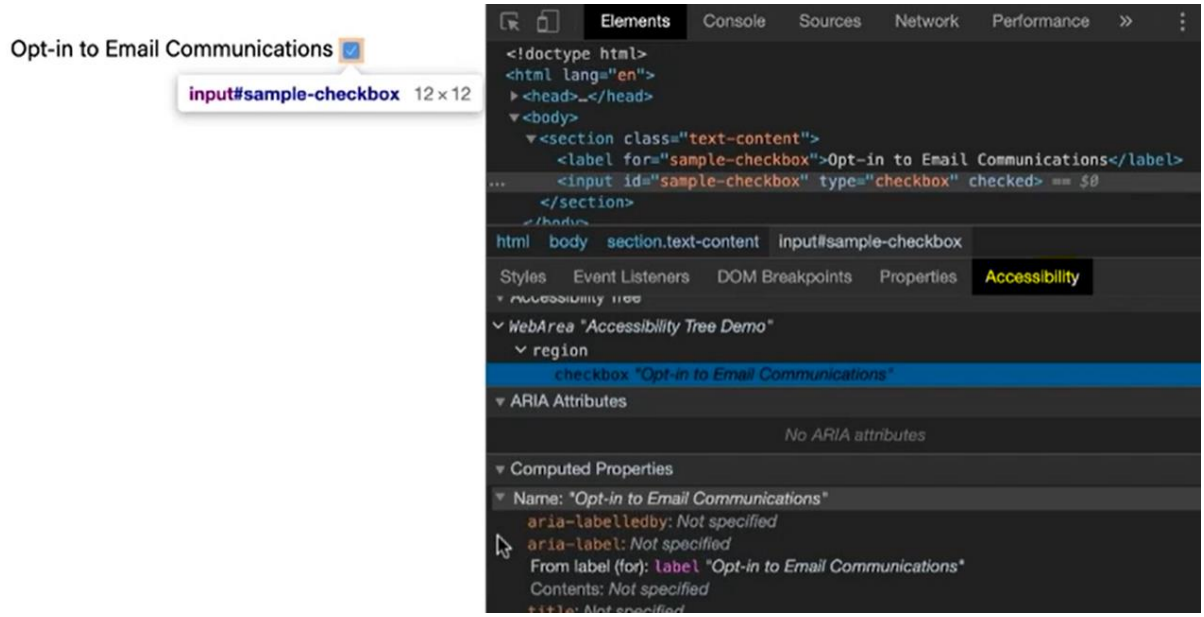
```
.skip-link {
    display: inline-block;
    padding: 5px 15px;
    background: white;
    z-index: 2;
    position: absolute;
    top: 0;
    left: -10000px;
}

.skip-link:focus {
    left: 0;
}
```



The NVDA is an example of screen reader.

Accessibility: Website Structure and Comprehension

Screen reader uses accessibility tree to guide user –



ARIA is a technical specification for improving the accessibility of web pages, it allows us to update the accessibility tree.

Common navigation patters are navigation bars, side navs, breadcrumb and hamburger menus

We need to use an hack to hide the content for the assistive technologies, because search engine and screen readers both ignore hidden content –

```
.visually-hidden {
    position: absolute;
    clip: rect(0 0 0 0);
    height: 1px;
    width: 1px;
    margin: -1px;
    padding: 0;
    border: 0;
    overflow: hidden;
}
```

Web Accessibility: Getting Started

Introduction

Web accessibility is to make websites and apps usable for everyone like for below:

- Not just those with disabilities
- People with vision impairments
- People using assistive technologies
- People with cognitive issues
- People with temporary issues
- People with changing abilities
- People in urgent situations

It will be hard and require more money to support accessibility if we try to implement it after project has been complete, if we start the project with accessibility in mind then it will become easy and without any additional budget.

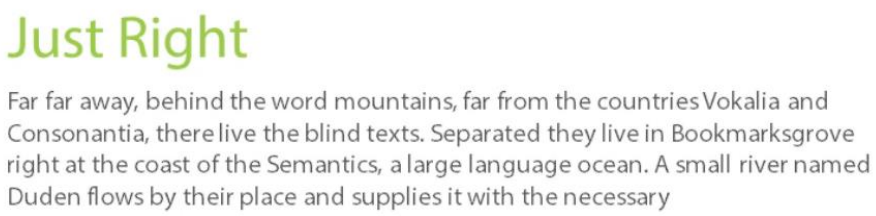
Designing for Accessibility

Responsive design is an accessible design practice. Like contrast level also for bright and dark environment.

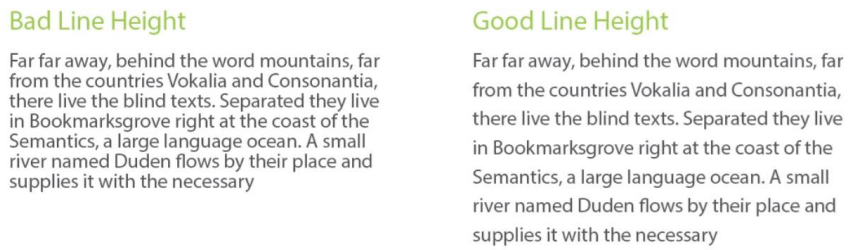
Color blind people have difficulty distinguishing between certain color combination like red/green deficiencies.

Typography – choosing right typeface and hierarchy of font sizes it should be in rage of 16px to 20px.

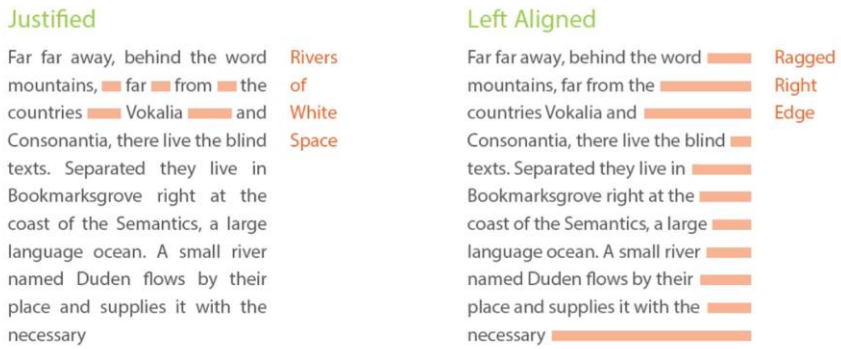
Don’t use too lengthy line or short line:



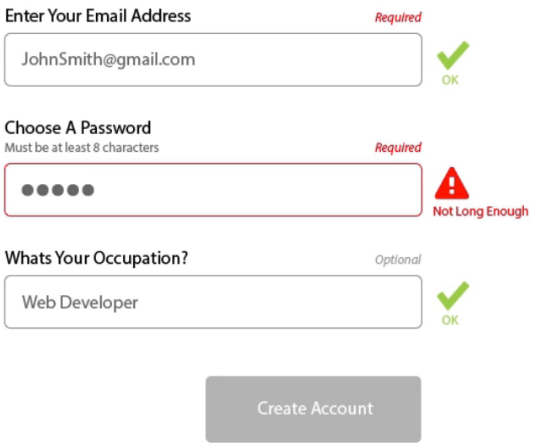
Good line height:



Use left aligned text instead of in-between justified:



Use accessible form - clearly identify required fields, using label, properly showing the error message, not just relying on error color border. Forms should be clear, organized and easy:



Make touch targets larger minimum size should be 44px to 48px. The bigger the better. Proper space between them.

Using focus and active indicators like showing currently active menu / tab.

Every transition or animation should have purpose. When everything is just flashing and sliding around all over the page, it is not good for anyone, it makes user sick. We should provide an option to the user to disable animations.

Designing the crisis – like where is the phone number, why is this page scrolling so crazy, I just need to talk to someone. Provide information for different failure cases.

Coding for Accessibility

Use description list when a description for each item is needed.

Facilitate keyboard navigation.

Images are inaccessible so we need to use alt attribute to provide description of content of the image.

For complex image like chart we can use longdesc attribute for long description text, it can also contain URL also:

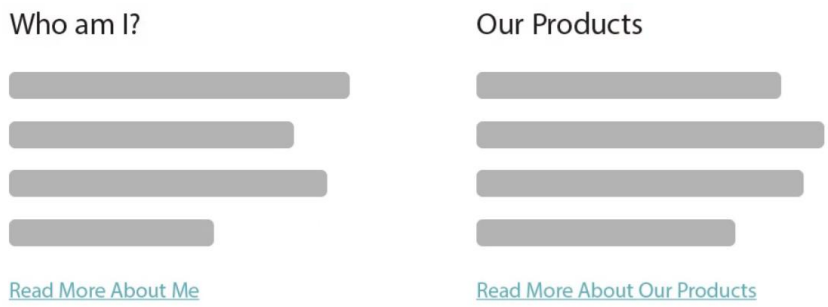
```
<img src="graph.jpg" longdesc="GraphDesc.txt" />
```

Or it is preferred to use aria-describedby attribute like below:

```
<img src="graph.jpg" aria-describedby="desc" alt="Graph" />
<a href="GraphDesc.html" id="desc">
   Get Graph Details
</a>
```

We should provide get transcript like for audio and video.

Instead of using “Read More” use a better description, don’t put it directly into anchor tag:



Instead of putting them inside anchor use CSS generated content “content” pseudo element:

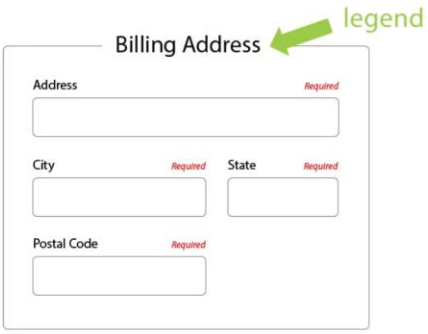
```
.about-link:after {
    content: “Read More”;
}
```

Provide lang attribute on HTML tag:

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

```
<blockquote lang=“fr">
    Si six scies scient six cyprès, six cents scies scient
    six cent cyprès.
</blockquote>
```

Use fieldsets and legends to logically group form fields;



We can use hidden labels but their text should exist we can make it visually hidden text in the code like in case of search box scenario, so that screen reader can read it:

```
<form id=“sitesearch”>
    <label for=“s” class=“hidden”>
        Search This Site
    </label>
    <input type=“text” id=“s” placeholder=“Search this site”>
    <input type=“button” value=“Go”>
</form>
```

```css
/* Visually Hide Content */
.hidden {
    position: absolute;
    left: -9999px;
}
```

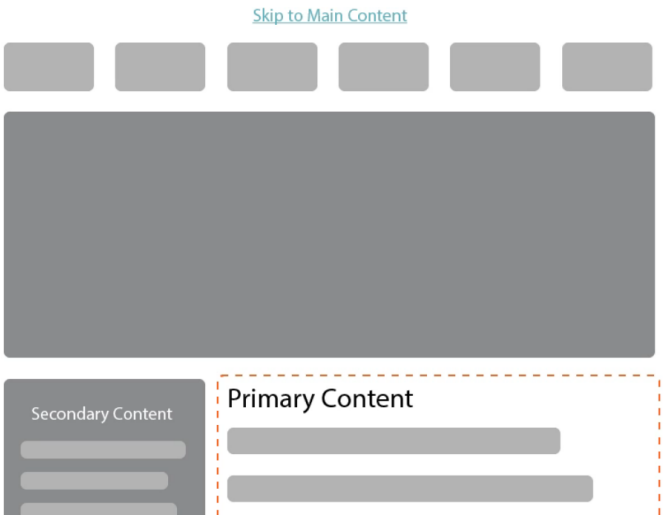
Using tab on non-standard controls like on span for force tab order for keyboard navigation:

```html
<span class="button" tabindex="5">
    Submit
</span>
```

Placeholder text should not be used as a replacement for labels, it should be used to provide hints and tips.

We should disable the mouse and check if we can control the website using keyboard only. Provide focus indicator, order of tab focus and in logical order. Who doesn't use a mouse:

if there are too many links before the main content then use a Skip to Main Content like before the navigation menu:



Use rem or em instead of pixel value for font size, so that if we increase it, it won't break the layout.

Use semantic elements.

Use visually hidden text by providing below CSS, so that it won't be display on the page, but screen reader will read it, like "Main Content" section:

```
.hidden {
    position: absolute;
    left: -9999px;
    top: auto;
    width: 1px;
    height: 1px;
    overflow: hidden;
}
```

Or, use CSS clip property:

```
.hidden {
    position: absolute;
    clip: 1px 1px 1px 1px; /* For ie6 & ie7 */
    clip: 1px, 1px, 1px, 1px;
}
```

There are below steps with Progressive Enhancement:

1. Start with the lowest common denominator
2. Well-structured HTML
3. Add in well-supported CSS
4. Add in more advanced CSS
5. Add in JavaScript to further enhance the experience

ARIA Roles & Attributes

ARIA is WIA-ARIA = web accessibility initiative – accessible rich internet applications. This attribute provides more accessibility information.

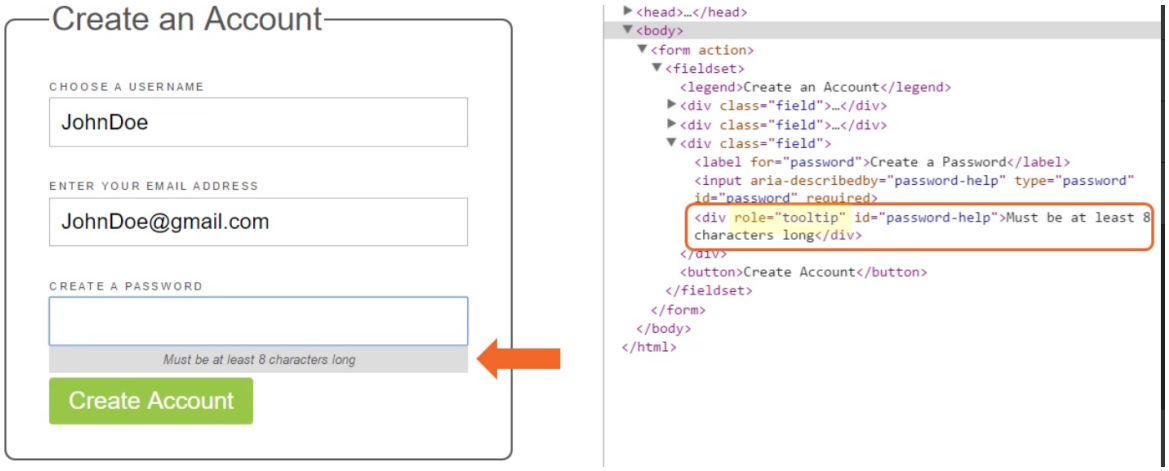
Aria roles – it defines what a widget is or what it does like menu, slider, progress meters, modals, or to describe structure – headings, regions, tables.

Aria sates defines the states like checked, disabled, selected, hidden, Aria properties are more related with characteristics of a widget like drag and drop, live regions and modals.

Aria roles – help navigate content, describe common regions like role="application", "banner", "navigation", "main", "search", "complimentary", "form", "contentinfo" (generally found in the footer). Banner, main and contentinfo role should be use once within a page.

Widget roles:

alert	menuitem	spinbutton
alertdialog	menuitemcheckbox	status
button	menuitemradio	tab
checkbox	option	tabpanel
dialog	progressbar	textbox
gridcell	radio	timer
link	scrollbar	tooltip
log	slider	treeitem
marquee		

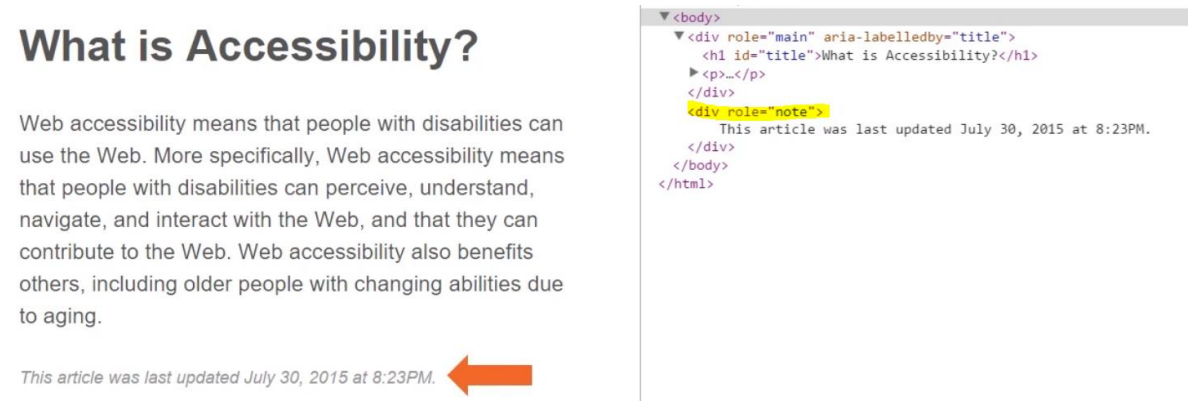


Container roles

combobox	menu	tablist
grid	menubar	tree
listbox	radiogroup	treegrid

Document structure roles – used to describe non-interactive elements that provides the structure of content.

article	img	region
columnheader	list	row
definition	listitem	rowgroup
directory	math	rowheader
document	note	separator
group	presentation	toolbar
heading		



Aria roles is about filing the gap between what is available and the semantics of the code that we are using and what's actually occurring in our rich internet applications.

ARIA states – describe dynamic states and changed with JavaScript:

aria-busy

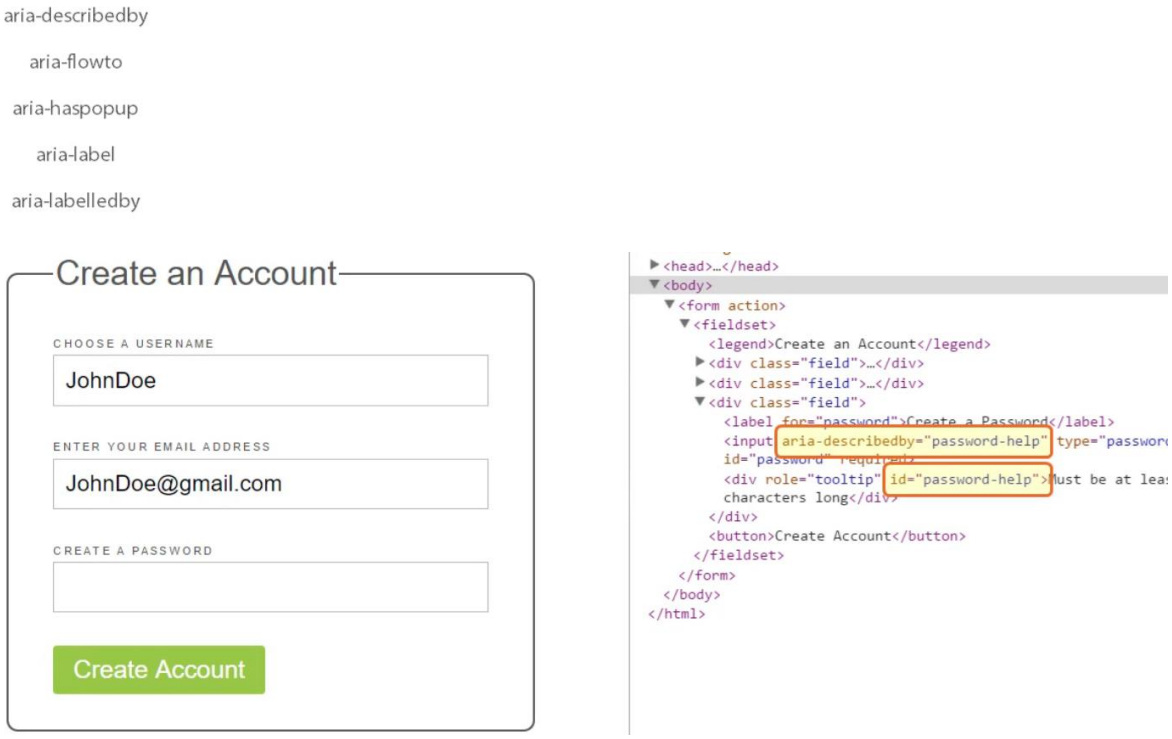
aria-disabled

aria-grabbed

aria-hidden

aria-invalid

Aria properties – they are primarily used to described relationships between elements and almost never change like states do.



For forms the spacebar should activates controls and the enter key should submit the default action of the form.

We should only use custom elements, widgets and ARIA when we either do not have an HTML equivalent control or when we absolutely cannot use the existing control because it doesn't have the functionality that we need.

Accessibility Testing, Tools, & Resources

Accessibility tools – test website by disabling the mouse, use screen reader, wave.webaim.org, colour contrast tool, accessibility developer tool by chrome extension, using audit tab from chrome developer debugger tool, tenon.io, a11yproject.com, the accessibility cheat sheet (http://bitsofco.de/2015/the-accessiblity-cheatsheet/), khan.github.io/tota11y/, a11ywins.tumblr.com, dylanb.github.io/periodic-area-roles.html