CSS Accessibility Review

Color Contrast Tools

- **WebAIM's Color Contrast Checker**: This online tool allows you to input the foreground and background colors of your design and instantly see if they meet the Web Content Accessibility Guidelines (WCAG) standards.
- **TPGi Colour Contrast Analyzer**: This is a free color contrast checker that allows you to check if your websites and apps meet the Web Content Accessibility Guidelines (WCAG) standards. This tool also comes with a Color Blindness Simulator feature which allows you to see what your website or app looks like for people with color vision issues.

Accessibility Tree

Accessibility tree is a structure used by assistive technologies, such as screen readers, to interpret and interact with the content on a webpage.

max() Function

The **max()** function returns the largest of a set of comma-separated values:

Example Code

```
img {
  width: max(250px, 25vw);
}
```

In the above example, the width of the image will be 250px if the viewport width is less than 1000 pixels. If the viewport width is greater than 1000 pixels, the width of the image will be 25vw. This is because 25vw is equal to 25% of the viewport width.

Best Practices with CSS and Accessibility

- **display: none;**: Using display: none; means that screen readers and other assistive technologies won't be able to access this content, as it is not included in the accessibility tree. Therefore, it is important to use this method only when you want to completely remove content from both visual presentation and accessibility.
- **visibility: hidden;**: This property and value hides the content visually but keeps it in the document flow, meaning it still occupies space on the page. These elements will

also no longer be read by screen readers because they will have been removed from the accessibility tree.

- **.sr-only CSS class**: This is a common technique used to visually hide content while keeping it accessible to screen readers.

Example Code

```
.sr-only {
 position: absolute;
 width: 1px;
 height: 1px;
 padding: 0;
 margin: -1px;
 overflow: hidden;
 clip: rect(0, 0, 0, 0);
 white-space: nowrap;
 border: 0;
}
```

- **scroll-behavior: smooth;**: This property and value enables a smooth scrolling behavior.
- **prefers-reduced-motion feature**: This is a media feature that can be used to detect the user's animation preference.

Example Code

```
@media (prefers-reduced-motion: no-preference) {
 * {
  scroll-behavior: smooth;
 }
}
```

In the above example, smooth scrolling is enabled if the user doesn't have animation preference set on their device.

Hiding Content with HTML Attributes

- **aria-hidden attribute**: Used to hide an element from people using assistive technology such as screen readers. For example, this can be used to hide decorative images that do not provide any meaningful content.
- **hidden attribute**: This attribute is supported by most modern browsers and hides content both visually and from the accessibility tree. It can be easily toggled with JavaScript.

Example Code

<p aria-hidden>This paragraph is visible for sighted users, but is hidden from assistive technology.</p>

<p hidden>This paragraph is hidden from both sighted users and assistive technology.</p>

Accessibility Issue of the placeholder Attribute

Using placeholder text is not great for accessibility. Too often, users confuse the placeholder text with an actual input value - they think there is already a value in the input.