Modern CSS Upgrades to Improve Accessibility

Stephanie Eckles (she/her)

@5t3ph • ModernCSS.dev

About Me

- ~15 years experience as a front-end focused developer
- Career journey: marketing, product, and design systems
- Writer, speaker, instructor, podcast host, mom, baker

Today we'll be learning…

Modern CSS capabilities for building accessibly inclusive layouts

Topics

01 Focus Visibility

02 Focus vs. Source Order

03 Desktop Zoom and Reflow

04 Respecting User Preferences

Focus Visibility

2.4.7: Focus Visible

Keyboard operable interfaces must have visible focus indicators

2.4.11: Focus Appearance (Minimum)

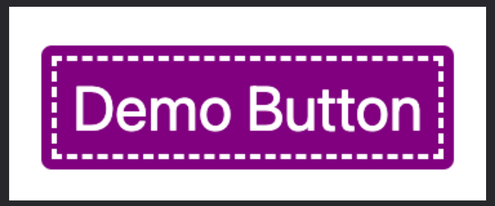
Draft in WCAG 2.2

Criteria for developing clearly distinguishable focus indicators

2.4.11: Focus Appearance (Minimum)

Any outline that is **at least 2px thick** and contrasts with the non-focused state would pass this criterion

Default appearance



Focused appearance

2.4.11: Focus Appearance (Minimum)

Minimum area

Outline

the area of a 1 CSS pixel thick perimeter of the unfocused component

or

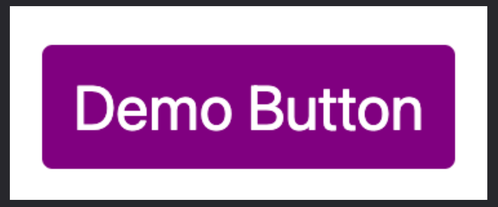
Shape

the area of a 4 CSS pixel thick line along the shortest side of a minimum bounding box of the unfocused component, and no thinner than 2 CSS pixels

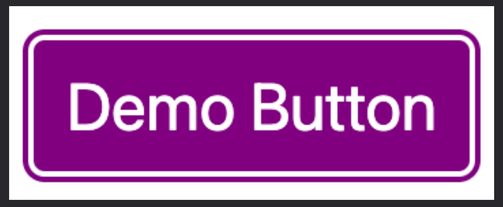
2.4.11: Focus Appearance (Minimum)

TL;DR for minimum area

Authors are encouraged to make the change as significant as possible, for example, by designing a thick border around the element



Default appearance



Focused appearance

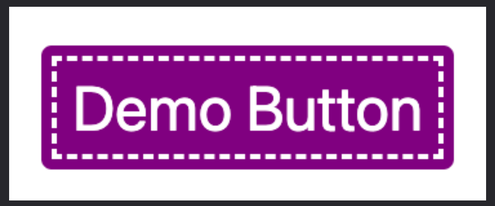
2.4.11: Focus Appearance (Minimum)

Contrasting area

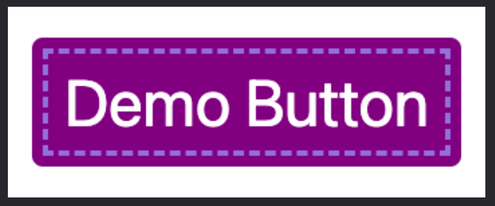
an area of the focus indicator **contrasts at least 3:1**

between the colors in the focused and unfocused states

Default appearance



Passes contrast



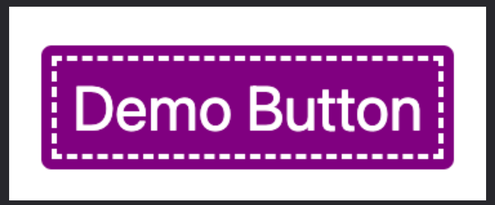
Fails contrast

2.4.11: Focus Appearance (Minimum)

Adjacent contrast

the contrasting area also has **a contrast ratio of least 3:1 against adjacent colors** in the focused component, or the contrasting area has a thickness of at least 2 CSS pixels

Default appearance



Passes adjacent contrast

Passes adjacent contrast

Ensuring Visible Focus with Modern CSS

Step 1:
Set outline properties on interactive elements

```
:is(a, button, input, textarea, summary) {
  --outline-size: max(2px, 0.08em);
  --outline-style: solid;
  --outline-color: currentColor;
}
```

Ensuring Visible Focus with Modern CSS

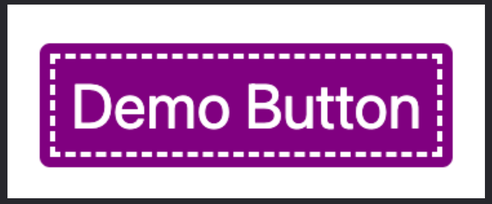
Step 2:
Apply outline properties on :focus

```
:is(a, button, input, textarea, summary):focus {
  outline:
    var(--outline-size)
    var(--outline-style)
    var(--outline-color);
  outline-offset: var(--outline-offset, var(--outline-size));
}
```

Ensuring Visible Focus with Modern CSS

Step 3:
Customize for specific elements/ components as needed

```
button {
  --outline-offset: -0.15em;
  --outline-style: dashed;
}
```



A note about :focus-visible

Based on heuristics, browsers by default may only show focus indicators for the state of :focus-visible

Meaning — possibly only keyboard users will see focus upon tabbing interactive elements if :focus is not defined

Topics

01 Focus Visibility

02 Focus vs. Source Order

03 Desktop Zoom and Reflow

04 Respecting User Preferences

Focus vs Source Order

2.4.3: Focus Order

For both visual and non-visual users, the focus order - which is typically initiated by keyboard tabbing - should proceed logically.

Usually this means matching source order to prevent visually jumping around randomly.

Link Link Link

Link Link Link

Link

7.
Link

5.
Link

1.
Link

4.
Link

6.
Link

2.
Link

3.
Link

Potential focus order breaking scenarios

Altering placement with

- absolute, fixed, or sticky positioning
- grid areas
- the order property for grid and flexbox
- masonry layout

How to fix focus order?

Be mindful of how you develop your source!

How to fix focus order?

Re-order the source instead of using CSS

Topics

01 Focus Visibility

02 Focus vs. Source Order

03 Desktop Zoom and Reflow

04 Respecting User Preferences

Desktop Zoom and Reflow

1.4.10 Reflow

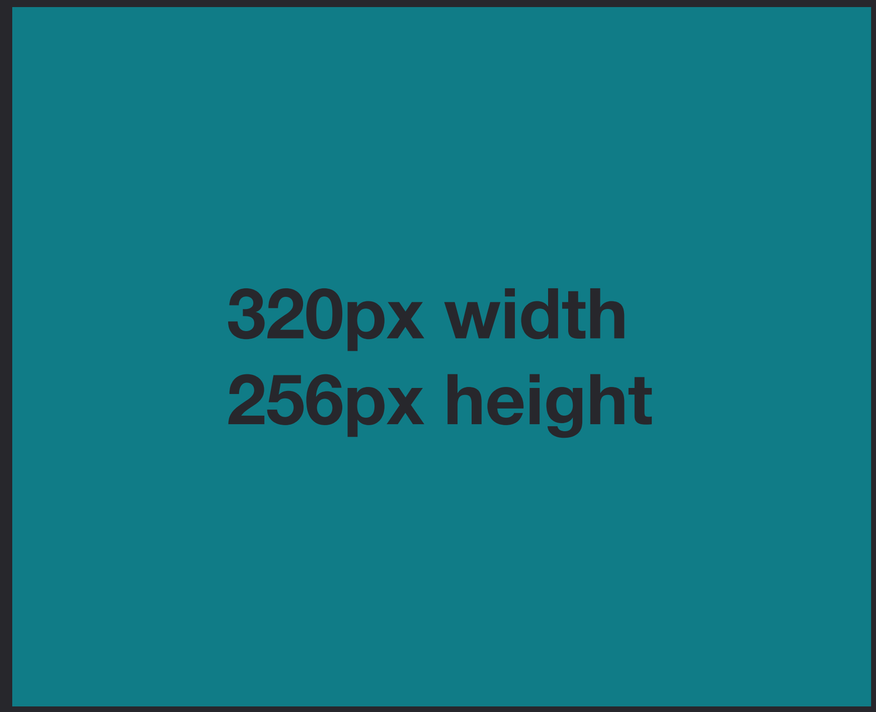
Reflow is the term for supporting desktop zoom up to 400%, where content should *reflow* into a single column, **without**:

- Loss of content or functionality
- Requiring scrolling in two dimensions

Desktop Zoom and Reflow

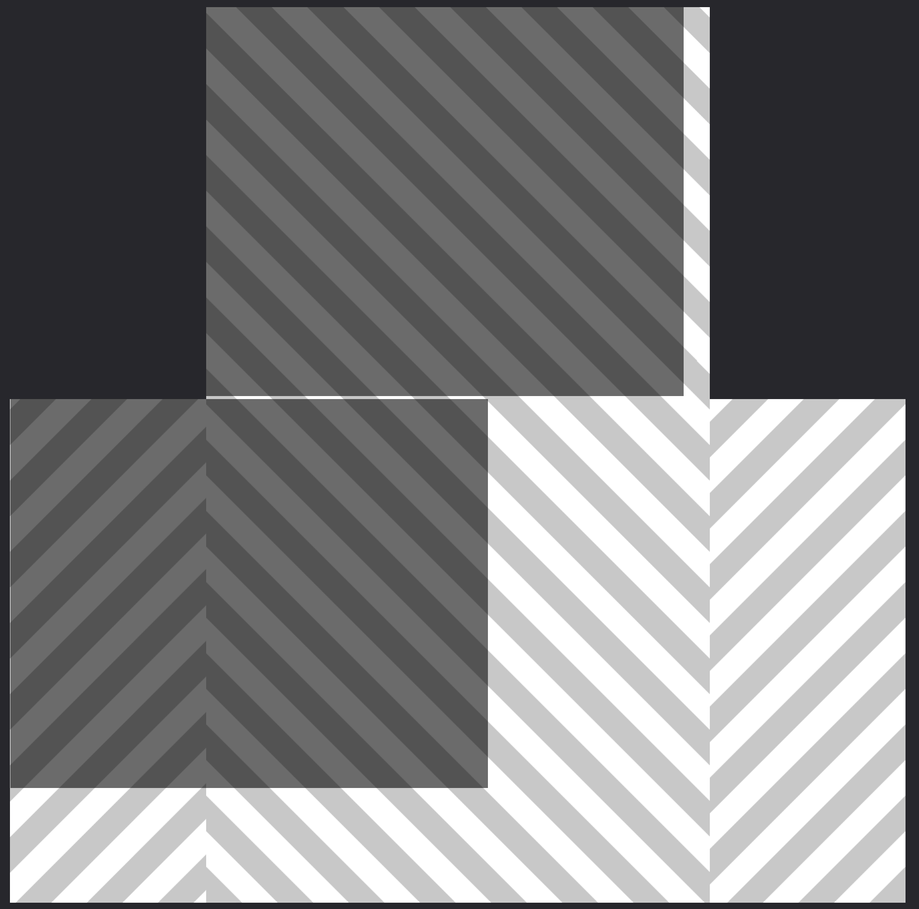
1.4.10 Reflow

On a 1280px wide resolution at 400%, the viewport content is equivalent to 320 CSS pixels wide



Desktop at 400% Zoom vs. iPhone SE

320px width
256px height



375px x 667px

Reflow vs. Responsive Design

Reframing expectations

- User is on a desktop, not a mobile device
- Re-arrange, do not remove, content and functionality
- Orientation is closer to landscape than portrait
- **Viewport size is not a proxy for device or user capabilities**

Media queries and reflow

There is no dedicated “zoom” media query

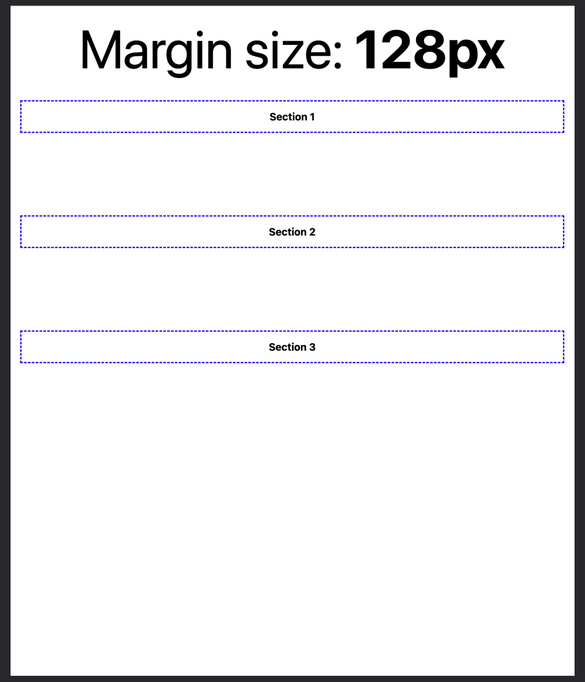
Media queries that affect viewports less than 320px will affect reflow

Potential reflow breaking scenarios

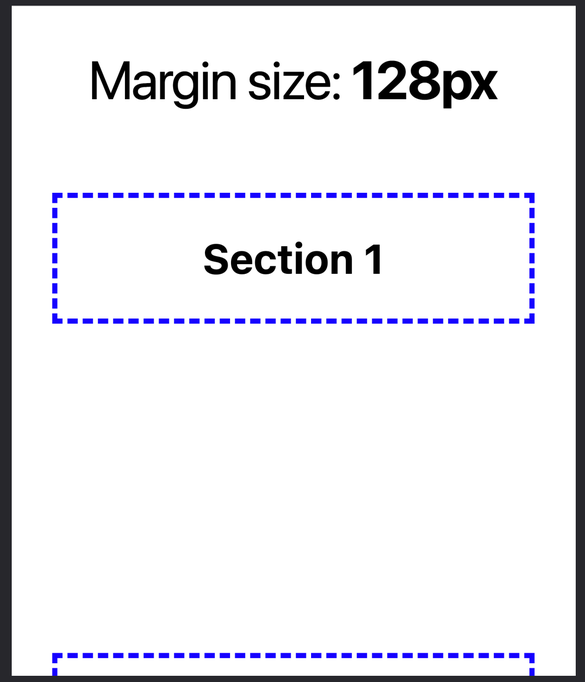
- sticky navigation that covers half or more of the viewport
- contained scroll areas become unscrollable/cut-off
- unwanted results when using fluid typography techniques
- overflow or overlap issues that cut-off content
- spacing appearing too large relative to the content size

Reflow and Section Spacing

```
section + section {
  margin-top: 128px;
}
```

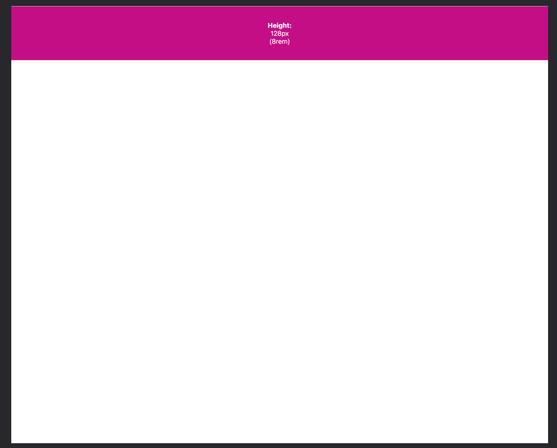


0% Zoom

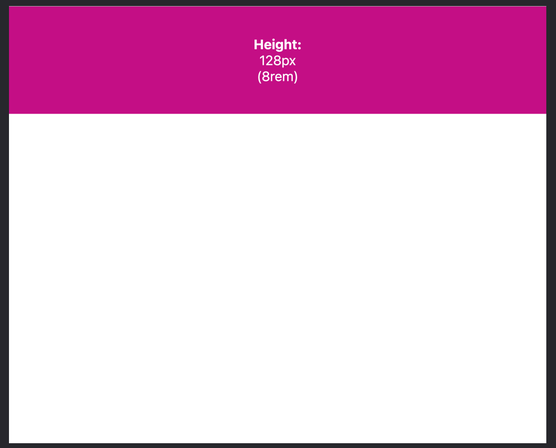


400% Zoom

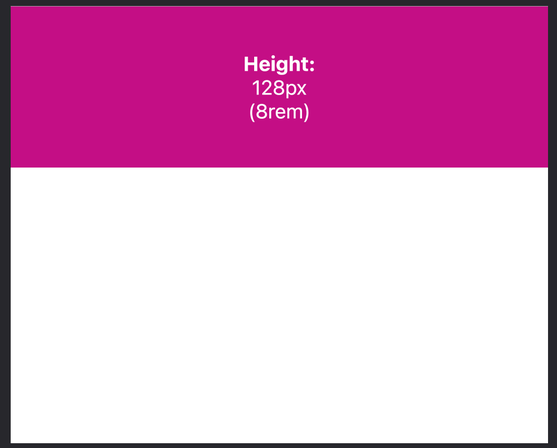
px/rem vs. vh



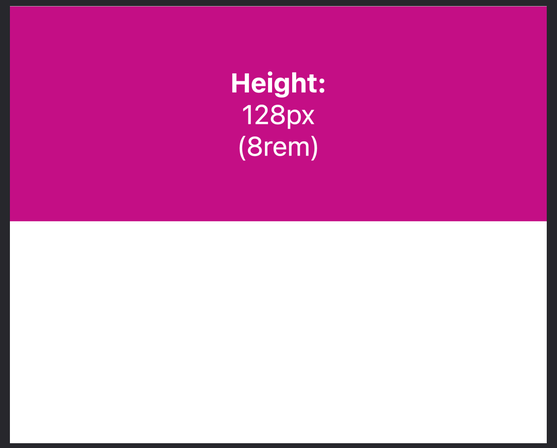
0% Zoom



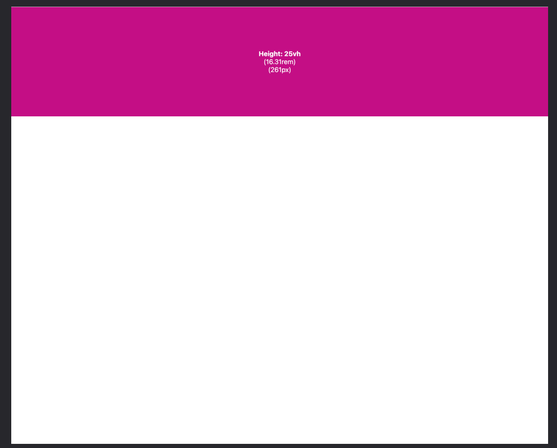
200% Zoom

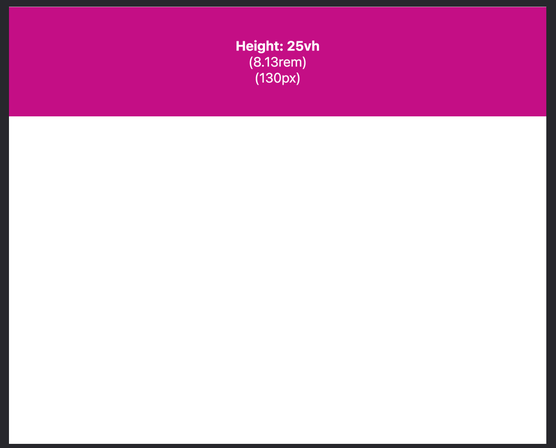


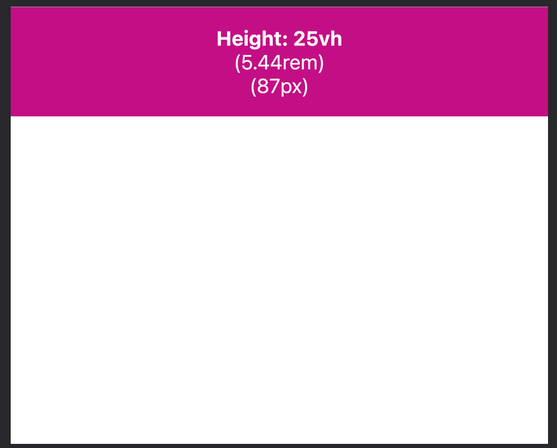
300% Zoom

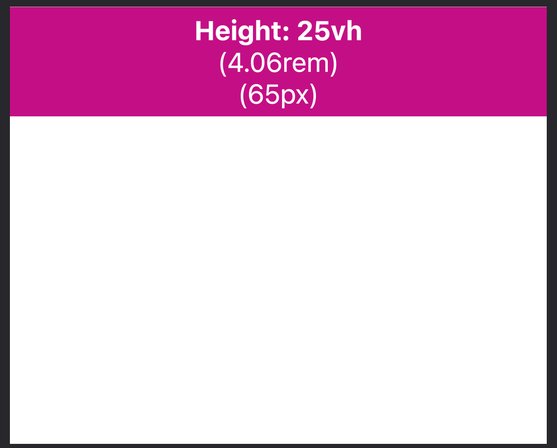


400% Zoom









px/rem vs. vh

Height: 25vh
(5.44rem)
(87px)

Viewport size: 426 x 347

300% Zoom

Height: 25vh
(4.06rem)
(65px)

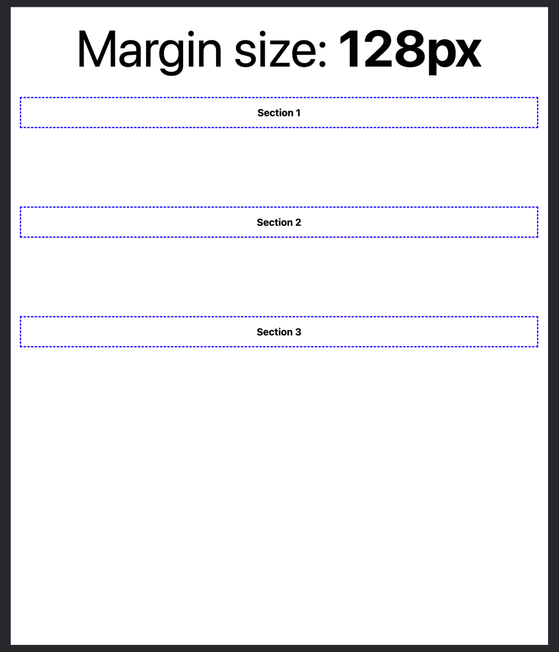
Viewport size: 320 x 260

400% Zoom

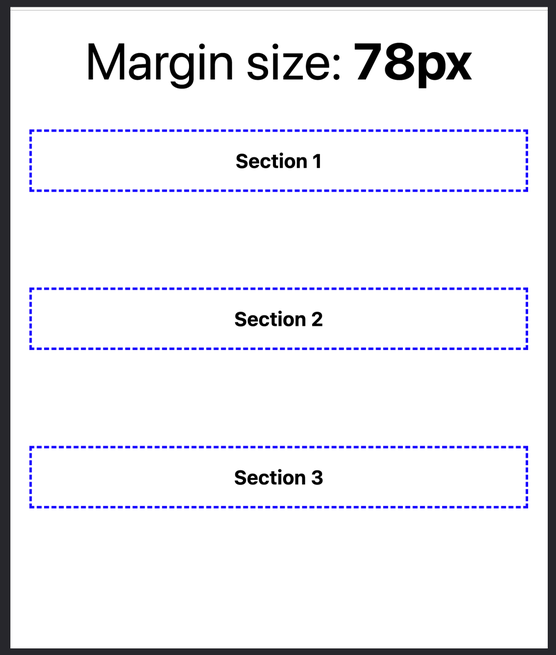
Modern CSS Section Spacing

```css
section + section {
  margin-top: 128px;
}
```

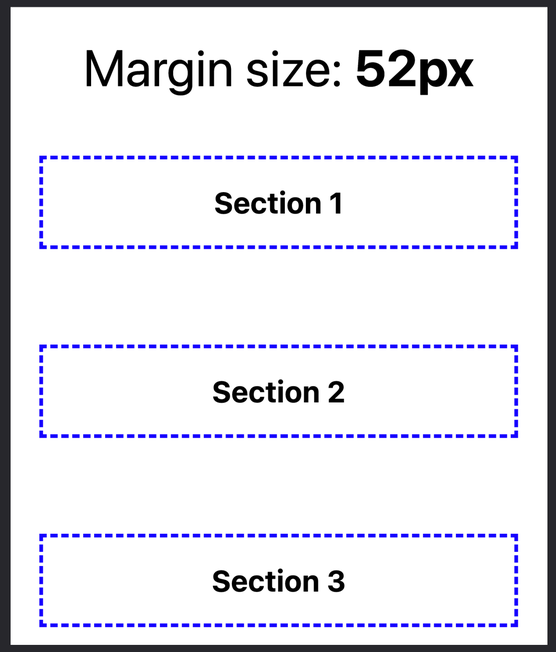
```css
section + section {
  margin-top: min(128px, 15vh);
}
```



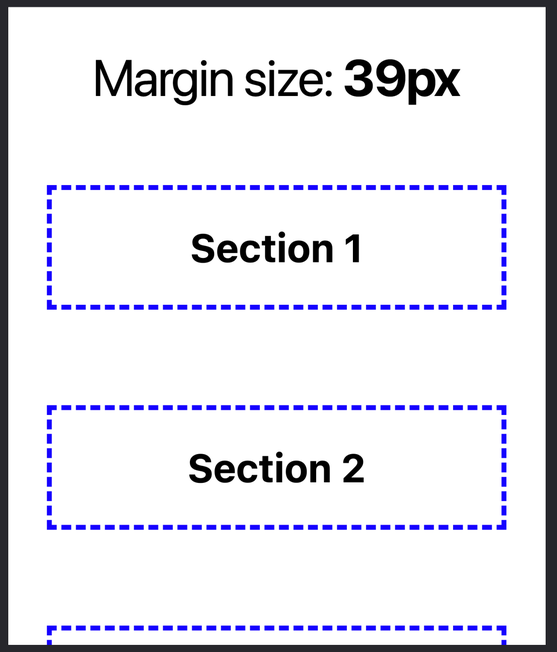
100% Zoom



200% Zoom



300% Zoom



400% Zoom

Modern CSS Element Padding

```
.card {
  padding: 1.5rem;
}
```

```
.card {
  padding: clamp(1rem, 5%, 1.5rem);
}
```

axe-con 2022

Lorem ipsum dolor sit amet consectetur adipisicing elit. Recusandae asperiores ipsam rem nisi fuga!

Sit amet

400% Zoom

Topics

01 Focus Visibility

02 Focus vs. Source Order

03 Desktop Zoom and Reflow

04 Respecting User Preferences

Respecting User Preferences

1. Motion
2. Color and contrast

Motion Criteria

2.3.1 Three Flashes or Below Threshold

Avoid anything that flashes more than three times in any one second period

2.3.3 Animation From Interactions

Motion animation triggered by interaction can be disabled, unless the animation is essential to the functionality or the information being conveyed

prefers-reduced-motion

- Detect operating system setting for motion preference
- Attach to feature query via CSS or JavaScript
- **Lack of setting does not mean user is ok with motion**

Motion reset

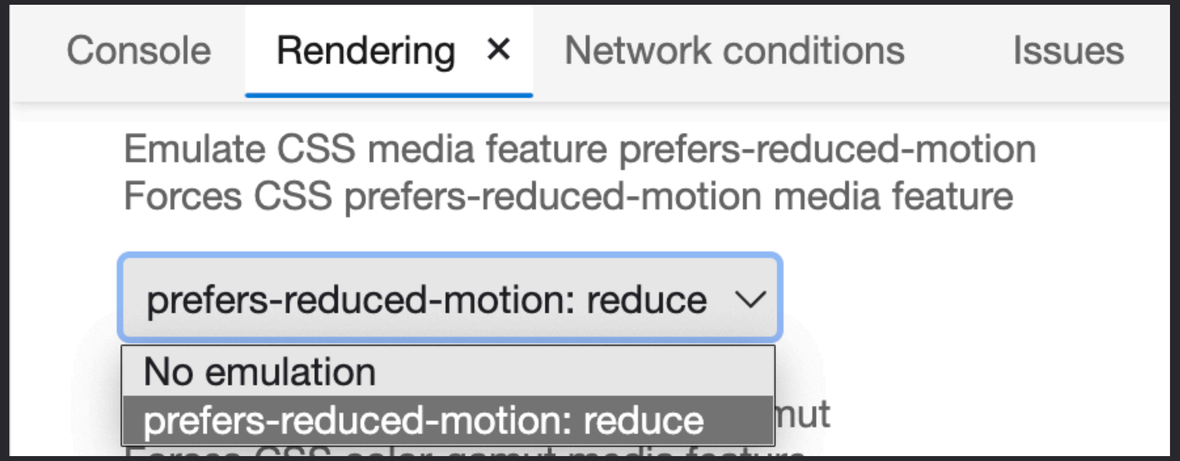
- Run all animations once, and complete transitions instantly
- Maintains duration for JavaScript events

```
@media (prefers-reduced-motion: reduce) {
  *,
  *::before,
  *::after {
    animation-duration: 0.01ms !important;
    animation-iteration-count: 1 !important;
    transition-duration: 0.01ms !important;
    scroll-behavior: auto !important;
  }
}
```

Credit: Andy Bell's Modern CSS Reset

Testing results of prefers-reduced-motion

In Chromium, found under More Tools > Rendering



Color and Contrast Criterion

1.4.3 Contrast Minimum

Provide enough contrast between text and its background so that it can be read by people with moderately low vision

No criteria currently indicate a requirement for “dark mode” or varying contrast modes

To respect dark/light mode and contrast modes is to **practice inclusive design**

Color and Contrast Feature Media Queries

1. prefers-color-scheme
2. prefers-contrast
3. forced-colors

All media queries adapt to
operating system preference

prefers-color-scheme

Explicitly define properties for “light” or “dark” color schemes

No requirement that “dark” is black, and “light” is white

```
@media (prefers-color-scheme: dark) {
  /* “dark" mode */
}

@media (prefers-color-scheme: light) {
  /* "light" mode */
}
```

Related property: color-scheme

```
:root {
  color-scheme: dark light;
}
```

```
<meta name="color-scheme"
content="dark light">
```

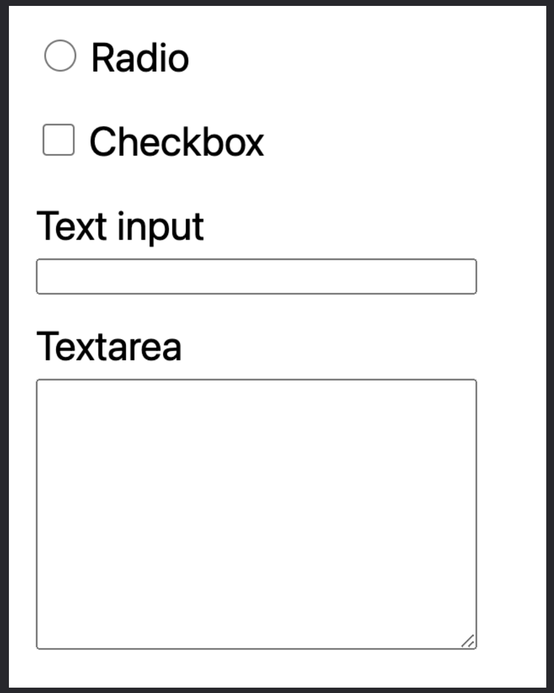
- Indicate a page supports light, dark, or both
- If set on :root or via meta tag, Chrome will auto-apply adjustments using system colors
- Order listed indicates preference

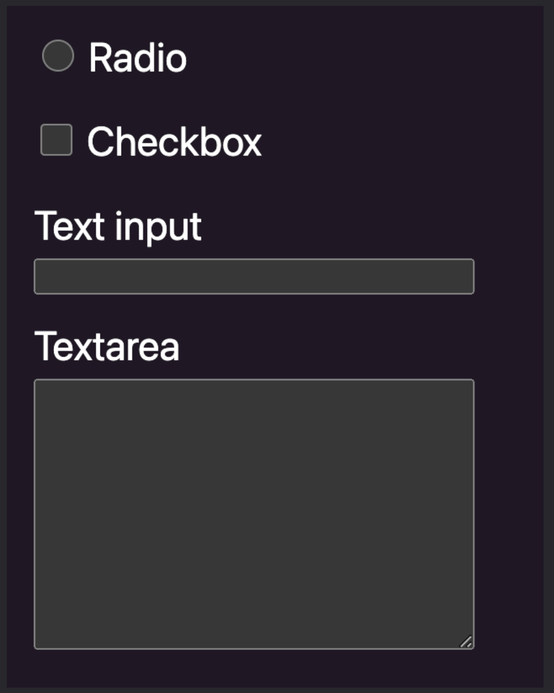
Related property: color-scheme

Alternatively, explicitly set for form controls only

```
input, select, textarea {
  color-scheme: light dark;
}

@media (prefers-color-scheme: dark) {
  body {
    background-color: #222;
    color: #fff;
  }
}
```





prefers-contrast

```
@media (prefers-contrast: no-preference) {}
@media (prefers-contrast: less) {}
@media (prefers-contrast: more) {}
@media (prefers-contrast: custom) {}
```

no-preference

Not set in operating system

custom

User defined contrast preference, implied if forced-colors query would match

prefers-contrast

⚠ **Not official guidance, more data needed**

“less”

Helps users with light sensitivity (photophobia), reduces migraine trigger

- Decrease text vs. background contrast
- Soften color contrast shifts between large areas
- Reduce brightness

“more”

Helps users read text & see details, distinguish UI, counter low-vision impairments (ex. Glaucoma)

- Increase text vs. background contrast
- Increase use/width of borders
- Remove box-shadows and other soft details

forced-colors

Increases text legibility through color contrast via built-in or user-defined color palettes

“active” means the user’s selected theme will overwrite your palette with system colors

```
@media (forced-colors: active) {}
@media (forced-colors: none) {}
```

forced-colors

Media query intent:

- Resolve colors for SVG icons
- Retain custom colors for critical features (ex. product color swatches)
- Resolve issues from lost color (ex. replace box-shadows with borders)

forced-colors

Removed/changed properties:

- box-shadow and text-shadow compute to none
- background-image computes to none unless the original value contains a url() function
- color-scheme computes to “light dark”
- scrollbar-color and accent-color computed to auto

forced-colors

Force-adjusted color properties:

- color
- fill
- stroke
- text-decoration-color
- text-emphasis-color
- border-color
- outline-color
- column-rule-color
- scrollbar-color
- -webkit-tap-highlight-color
- background-color
- caret-color
- flood-color
- lighting-color
- stop-color

forced-colors

System colors

- Canvas *and* CanvasText
- LinkText, VisitedText, *and* ActiveText
- ButtonFace, ButtonText, ButtonBorder
- Field *and* FieldText
- Highlight *and* HighlightText
- SelectedItem *and* SelectedText
- Mark *and* MarkText
- GreyText

Authoring for Feature Queries

prefers-color-scheme

Provide darker and lighter versions that still fully use brand colors and high-fidelity visuals

prefers-contrast

Provide “more” and “less” contrast versions that may require modified palettes and assets

forced-colors

Only use to correct for loss or change of color in critical elements

prefers-contrast vs. forced-colors

prefers-contrast

User still wants to see your design and colors,
but adjusted to the contrast preference

forced-colors

User requires using their own color palette
for improved usability

Authoring for Feature Queries

Chain feature queries when needed

```
@media (prefers-color-scheme: dark) and
(prefers-contrast: more) {}
```

```
@media (prefers-color-scheme: light) and
(prefers-contrast: less) {}
```

Topics

01 Focus Visibility

02 Focus vs. Source Order

03 Desktop Zoom and Reflow

04 Respecting User Preferences

Setup consistent, customizable :focus styles using custom properties

Topics

01 Focus Visibility

02 Focus vs. Source Order

03 Desktop Zoom and Reflow

04 Respecting User Preferences

Learned about order-breaking properties and to change order in the source

Topics

01 Focus Visibility

02 Focus vs. Source Order

03 Desktop Zoom and Reflow

04 Respecting User Preferences

Traded px for vh and %, using CSS functions for dynamic, contextual spacing

Topics

01 Focus Visibility

02 Focus vs. Source Order

03 Desktop Zoom and Reflow

04 Respecting User Preferences

Considered feature queries and their benefits for inclusive design

Modern CSS Upgrades to Improve Accessibility

Stephanie Eckles (she/her)
@5t3ph • ModernCSS.dev

Demos & Links

ModernCSS.dev/axecon22