

# Web Accessibility: it's not *\*just\** about HTML

Ire Aderinokun @  
ffconf 2023

@ireaderinokun / @ire@front-end.social

**“By default, HTML is accessible.”**

Web accessibility involves ensuring that content remains accessible.”

<https://developer.mozilla.org/en-US/docs/Learn/Accessibility>

Web accessibility is the **inclusive practice** of ensuring there are no barriers that prevent access to websites

# Physical disabilities



# Situational disabilities



# Socio-economic restrictions



Accessibility is about **inclusion**





# WCAG

W3C Accessibility  
Guidelines



1. Content must be **perceivable** 🧠

2. Interface must be **operable** 🛠️

3. Content must be **understandable** 📖

4. Code must be **robust** ⚙️

“By default, HTML is accessible.

Web accessibility involves ensuring that content remains accessible.”

<https://developer.mozilla.org/en-US/docs/Learn/Accessibility>

“By default, HTML is accessible, **if used correctly.**”

Web accessibility involves ensuring that content remains accessible.”

And it's 🦄ing perfect.

## Seriously, what the 🦄 else do you want?

You probably build websites and think your 🦄 is special. You think your 13 megabyte parallax-ative home page is going to get you some 🦄ing Awwward banner you can glue to the top corner of your site. You think your 40-pound jQuery file and 83 polyfills give IE7 a 🦄 because it finally has box-shadow. Wrong, mother🦄er. Let me describe your perfect-🍌 website:

- 🦄's lightweight and loads fast
- Fits on all your 🦄ty screens
- Looks the same in all your 🦄ty browsers
- The mother🦄er's accessible to every 🍌hole that visits your site
- 🦄's legible and gets your 🦄ing point across (if you had one instead of just 5mb pics of hipsters drinking coffee)

## Well guess what, mother🦄er:

You. Are. Over-designing. Look at this 🦄. It's a mother🦄ing website. Why the 🦄 do you need to animate a 🦄ing trendy-🍌 banner flag when I hover over that useless piece of 🦄? You spent hours on it and added 80 kilobytes to your 🦄ing site, and some mother🦄er jabbing at it on their iPad with fat sausage fingers will never see that 🦄. Not to mention blind people will never see that 🦄, but they don't see any of your 🦄ty 🦄.

You never knew it, but this is your perfect website. Here's why.

## It's 🦄ing lightweight

This entire page weighs less than the gradient-meshed facebook logo on your 🦄ing Wordpress site. Did you seriously load 100kb of jQuery UI just so you could animate the 🦄ing background color of a div? You loaded all 7 fontfaces of a 🦄ty webfont just so you could say "Hi " at 100px height at the beginning of your site? You piece of 🦄.

100

## Accessibility

These checks highlight opportunities to [improve the accessibility of your web app](#). Only a subset of accessibility issues can be automatically detected so manual testing is also encouraged.

ADDITIONAL ITEMS TO MANUALLY CHECK (10)

Show

These items address areas which an automated testing tool cannot cover. Learn more in our guide on [conducting an accessibility review](#).

PASSED AUDITS (16)

Show

NOT APPLICABLE (28)

Show

📅 Captured at Feb 25, 2022,  
1:59 PM GMT

🖥️ Emulated Moto G4 with  
Lighthouse 9.1.0

🔗 Single page load

🕒 Initial page load

🚦 Slow 4G throttling

🔗 Using Chromium  
98.0.4758.109 with  
devtools

Generated by [Lighthouse 9.1.0](#) | [File an issue](#)

HTML ***\*can\**** be inaccessible

# Missing text alternatives 🙄

```

```

# Using the wrong elements 😞

```
<span>Enter your username:</span>
```

```
<input type="text">
```

# Weird tab order 😞

```
<ul>
```

```
  <li><a href="/one" tabindex="2">Link One</a></li>
```

```
  <li><a href="/two" tabindex="3">Link Two</a></li>
```

```
  <li><a href="/three" tabindex="1">Link Three</a></li>
```

```
</ul>
```





**HTML**



# Adding content with CSS 😞

```
<div id="important"></div>
```

```
#important::after {
```

```
  content: "Really important information that everyone should know"
```

```
}
```

# Altering element behaviour with JavaScript 🙄

```
<div onClick="goToPage( '/about' )">
```

About Us

```
</div>
```

90% of accessibility is about **using HTML correctly.**

The other 10% is about **not using CSS/JS incorrectly.**

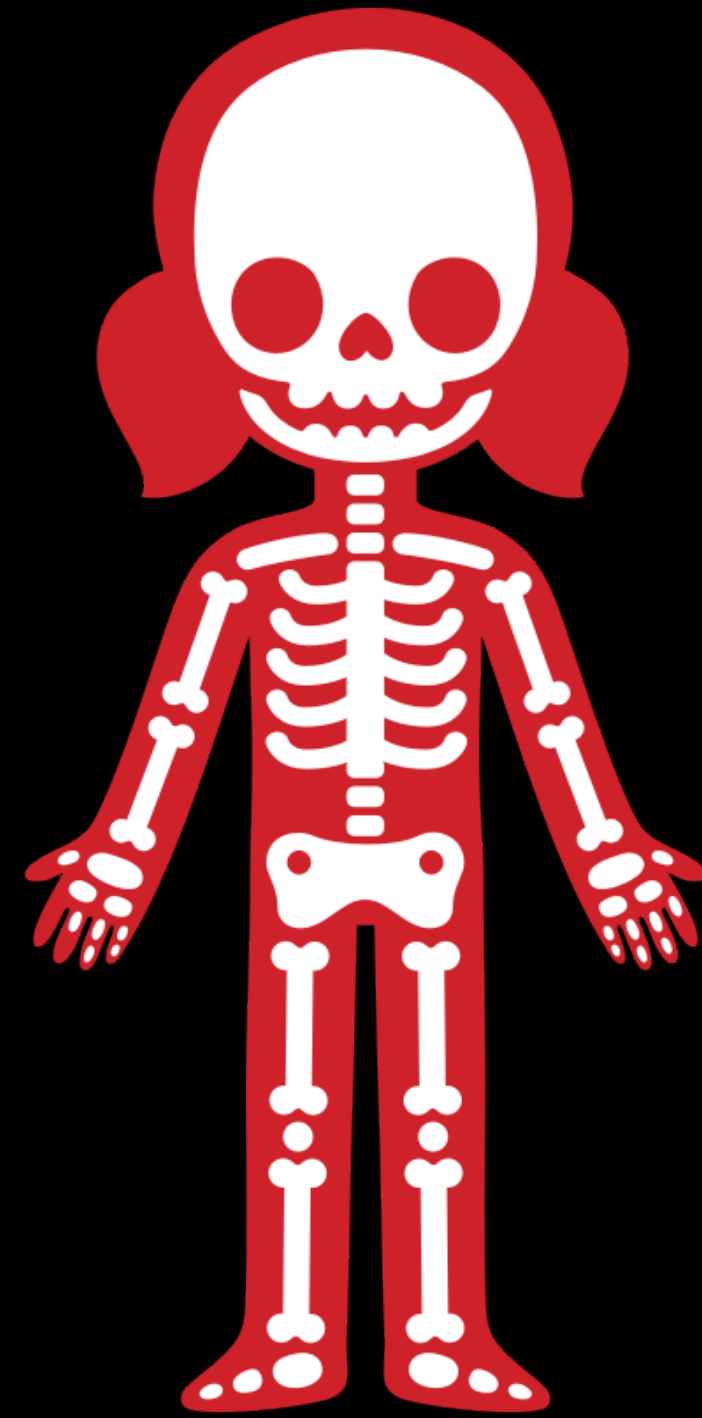
# CSS, JavaScript, & Accessibility

Part One

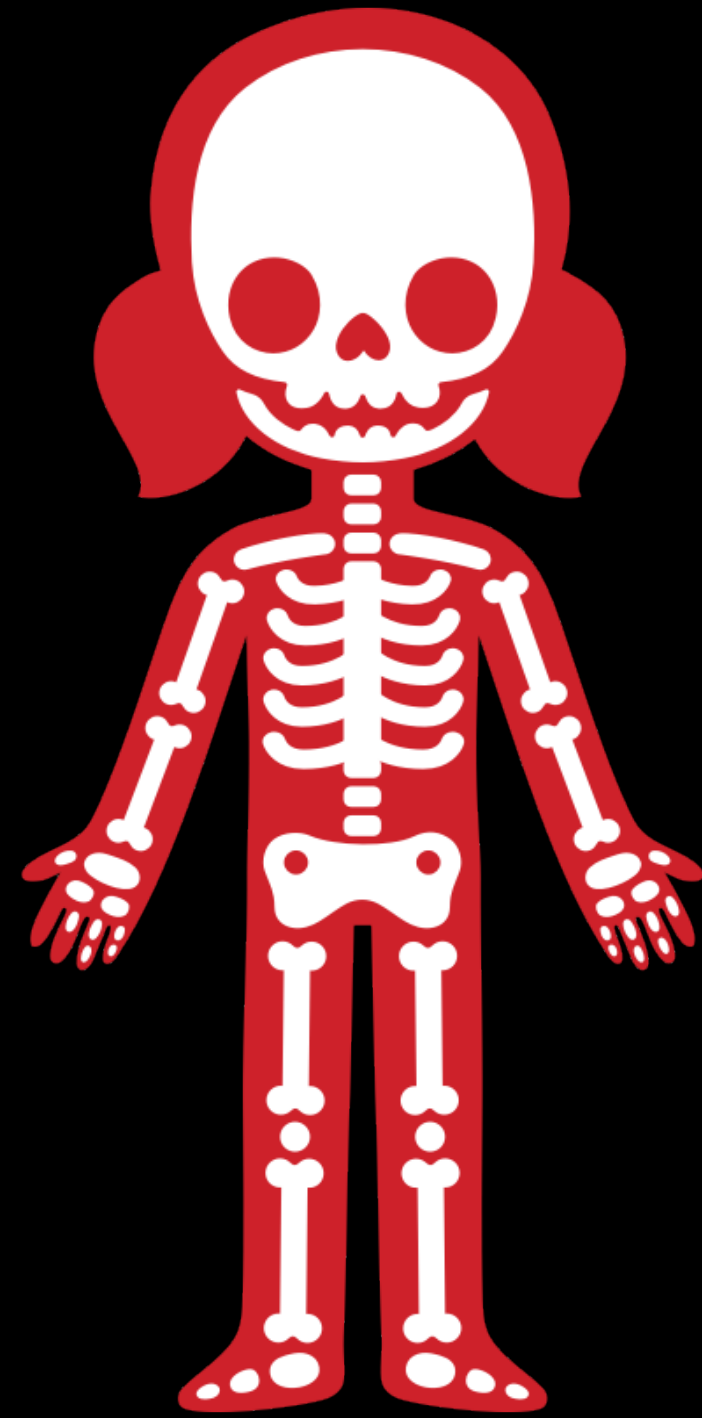
# CSS & Accessibility

CSS is used to **describe the presentation**  
of an HTML document





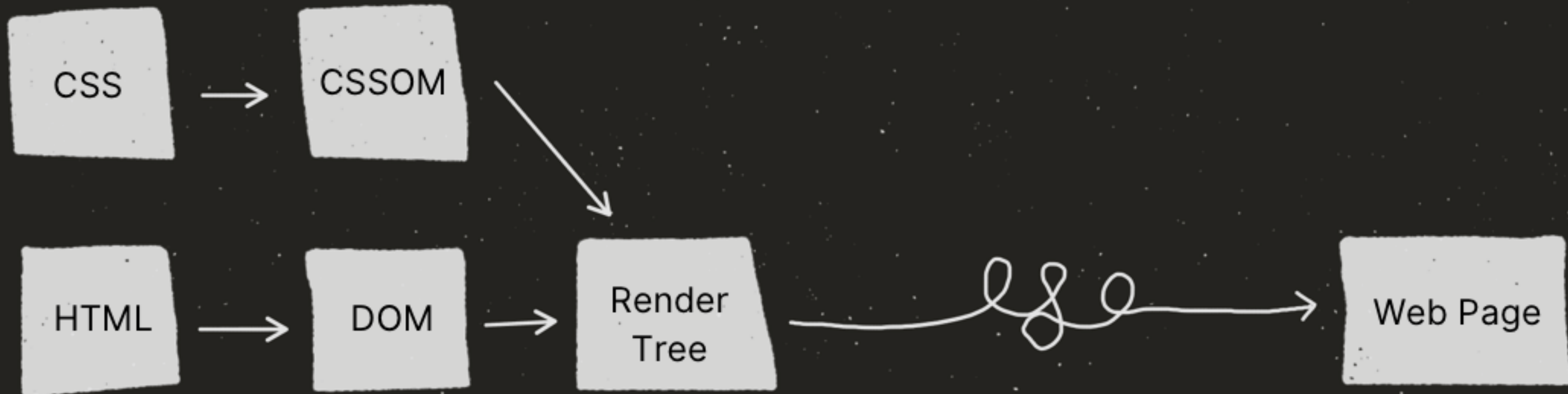
HTML



HTML

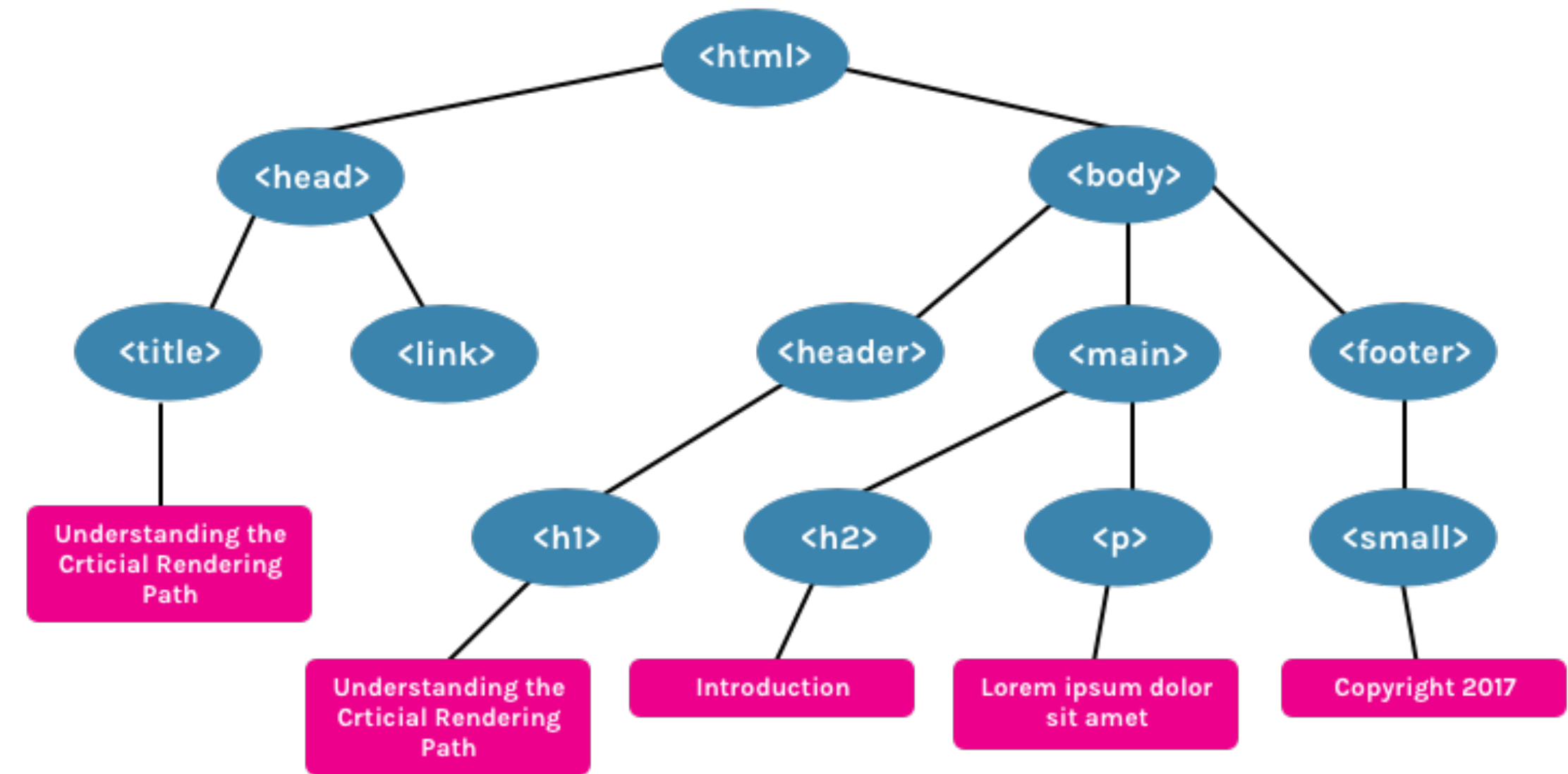


CSS



# DOM

```
<head>
  <title>Understanding the Critical Rendering Path</title>
  <link rel="stylesheet" href="style.css">
</head>
<body>
  <header>
    <h1>Understanding the Critical Rendering Path</h1>
  </header>
  <main>
    <h2>Introduction</h2>
    <p>Lorem ipsum dolor sit amet</p>
  </main>
  <footer><small>Copyright 2017</small></footer>
</body>
```



# CSSOM

```
body { font-size: 18px; }
```

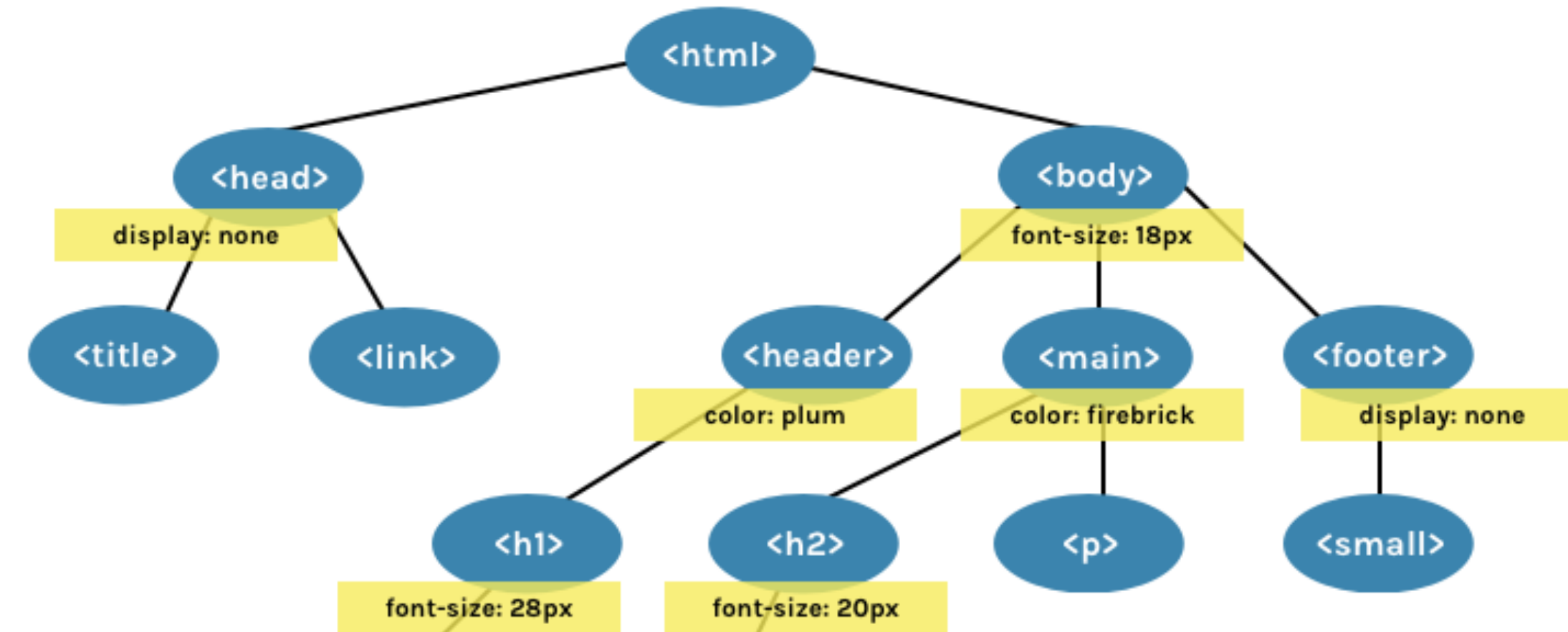
```
header { color: plum; }
```

```
h1 { font-size: 28px; }
```

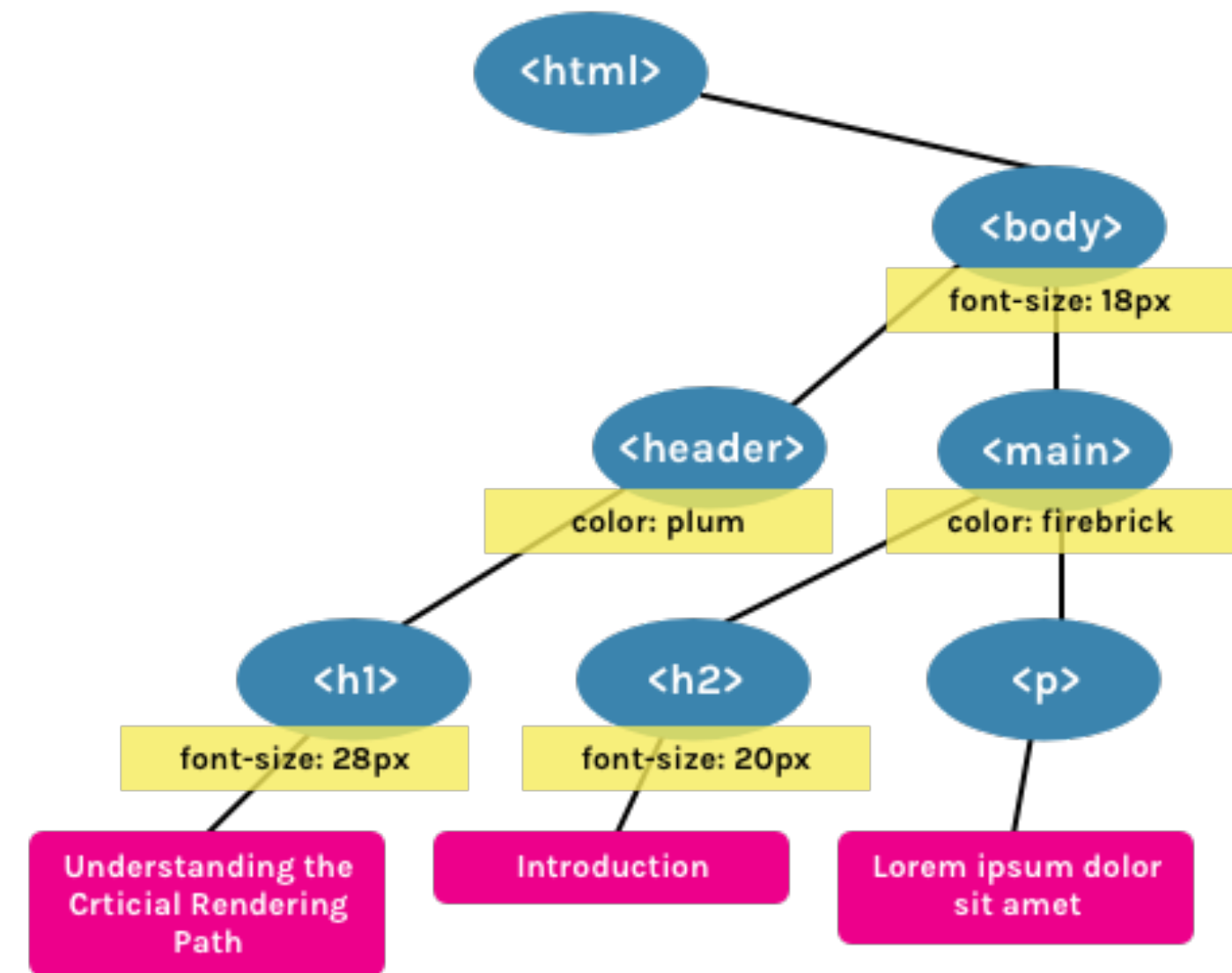
```
main { color: firebrick; }
```

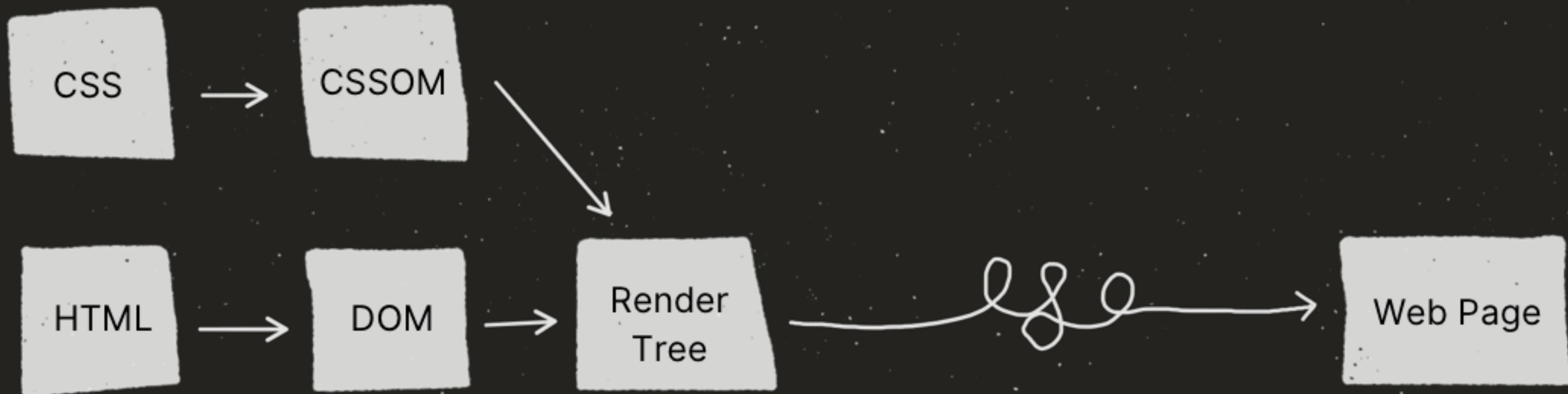
```
h2 { font-size: 20px; }
```

```
footer { display: none; }
```



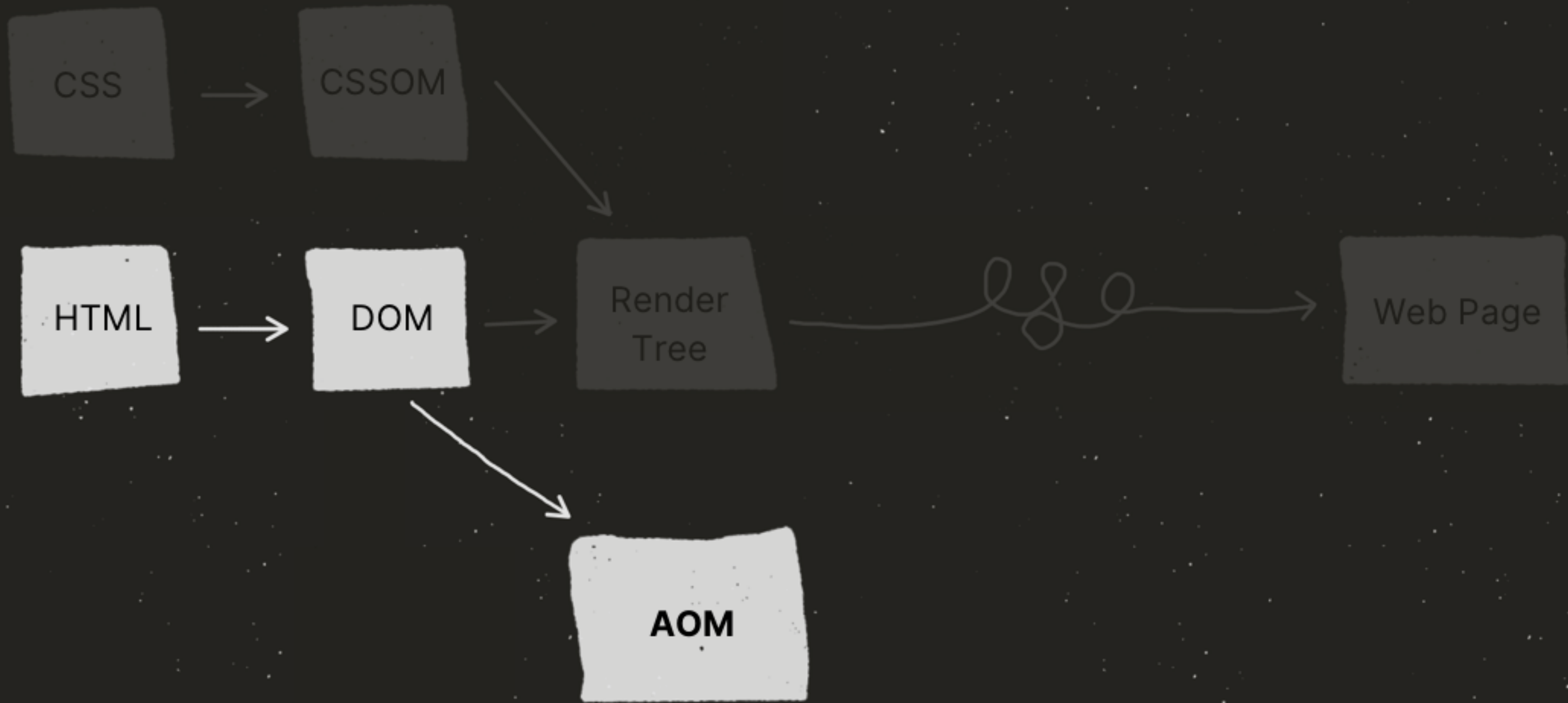
**DOM**  
**+ CSSOM**  
**- Non-visible elements**  
**= Render tree**





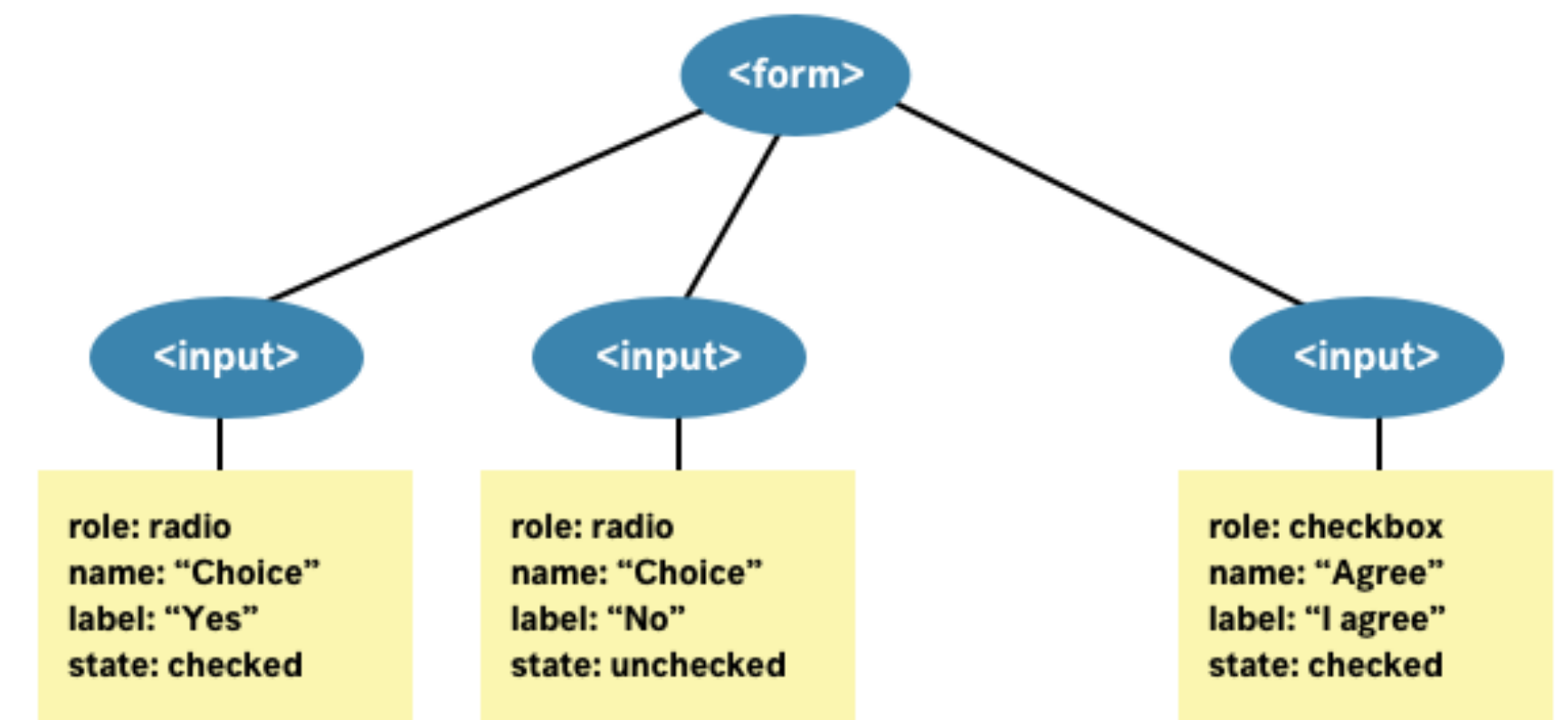






# AOM

```
<form>  
  
  <p>Make a choice:</p>  
  
  <input type="radio" name="choice" id="yes" />  
  <label>Yes</label>  
  
  <input type="radio" name="choice" id="no" />  
  <label>No</label>  
  
  
  <input type="checkbox" name="agree" id="agree" />  
  <label>I agree</label>  
  
</form>
```



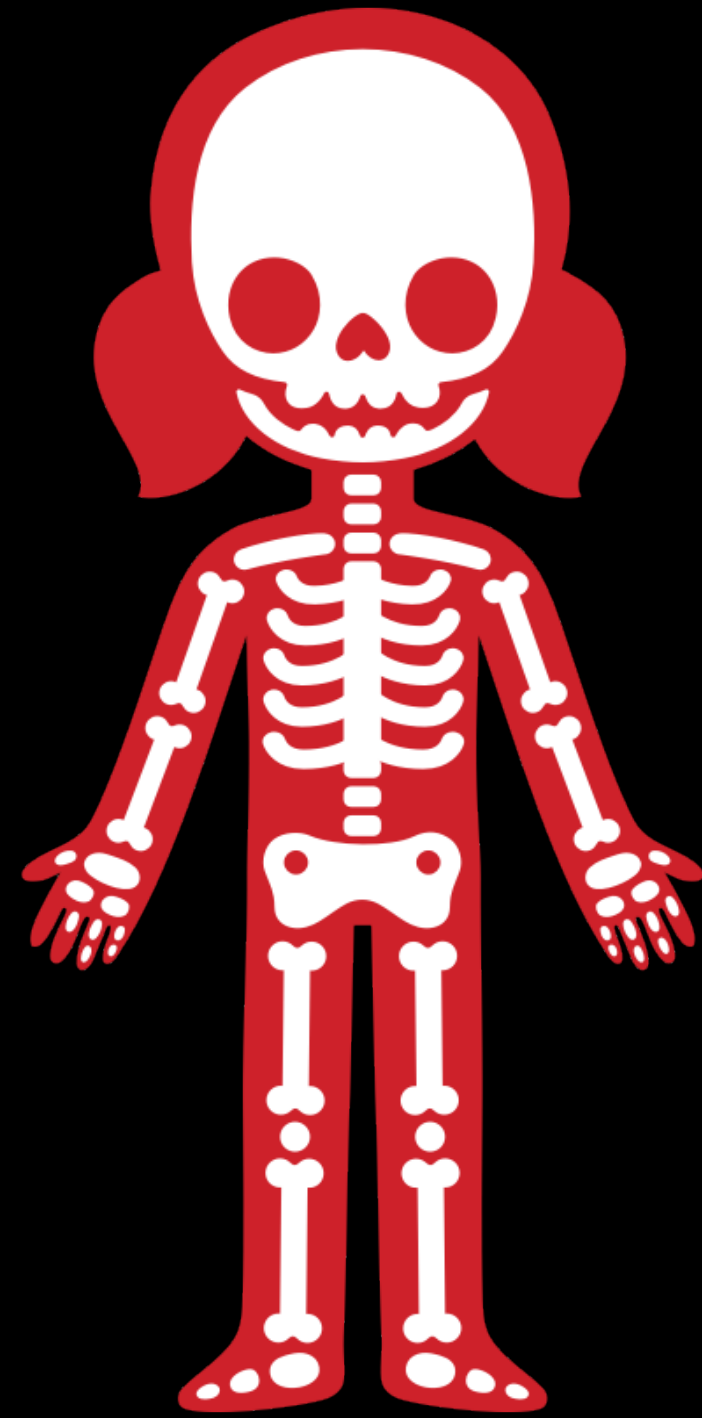
Search or enter website name

# Form

Make a choice:

Yes  No

I agree



HTML



CSS

#1

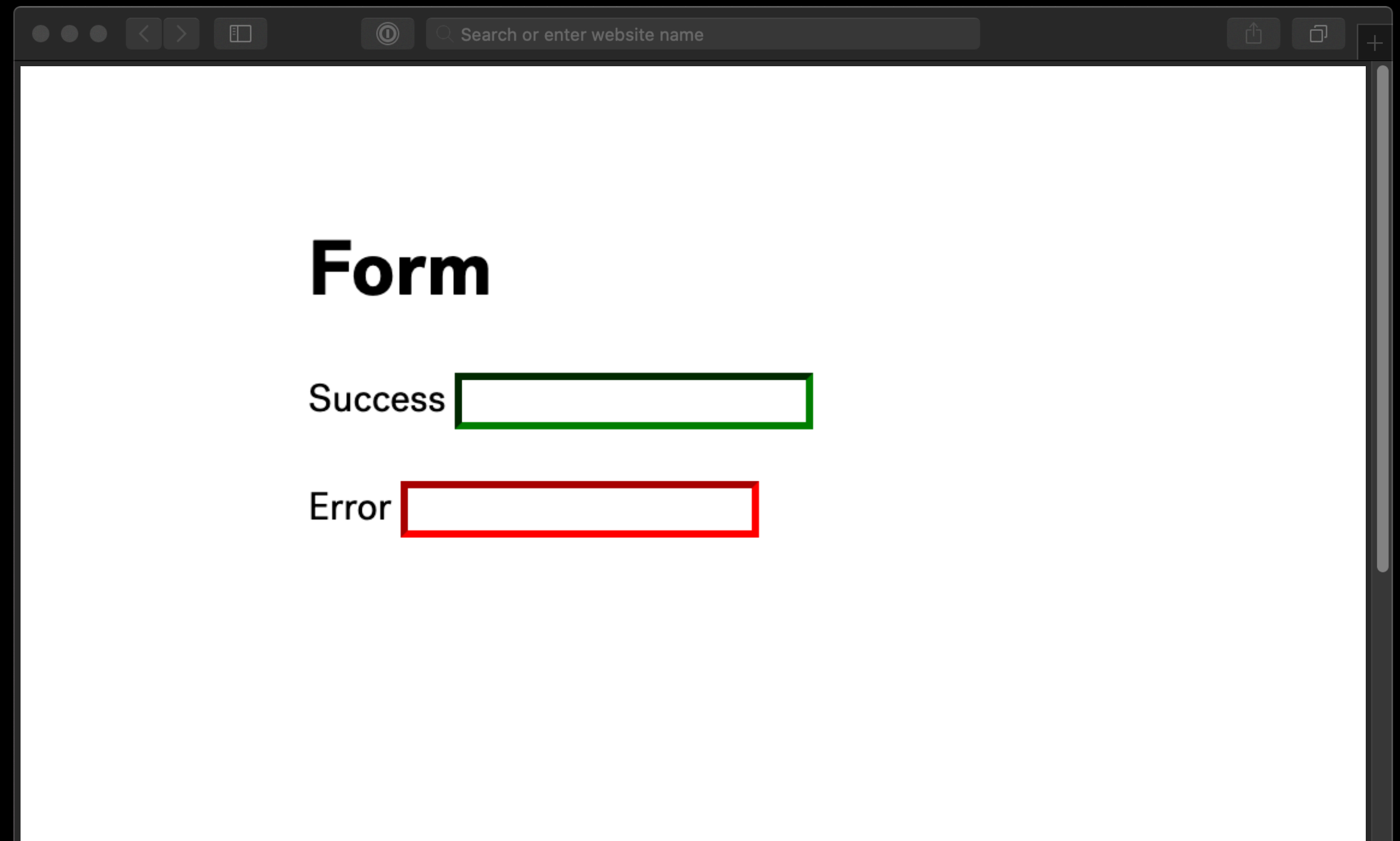
**Don't** use CSS to convey meaning or content

```
<input class="error" >
```

```
<input class="success" >
```

```
.error { border-color: red; }
```

```
.success { border-color: green; }
```



```
h1::after {
```

```
  content: "My Page Title"
```

```
}
```

## Browser and screen reader support for CSS generated content

	<b>Chrome 41 (Android)</b>	<b>Chrome 41 (Windows)</b>	<b>Firefox 36 (Windows)</b>	<b>Internet Explorer 11 (Windows)</b>	<b>Safari 8 (OSX)</b>	<b>Safari 8.1 (iOS)</b>
Jaws 16	N/A	Yes	Yes	No	N/A	N/A
NVDA 2015.1	N/A	Yes	Yes	No	N/A	N/A
TalkBack	Yes	N/A	N/A	N/A	N/A	N/A
VoiceOver	N/A	N/A	N/A	N/A	Yes	Yes

With Internet Explorer accounting for about 15% of traffic (in March 2015), there is good reason to consider the viability of using CSS generated content.



“In other words, use CSS generated content to change or supplement the design, **but not to create or alter important content** on the page.”

— Léonie Watson

<https://tink.uk/accessibility-support-for-css-generated-content/>

#2

**Don't** use CSS to change the  
semantics of HTML



```
display: none;      visibility: hidden;      opacity: 0;      position: absolute;
                    top: -9999px;
                    left: -9999px;
```

---

**Visually hidden?**



`display: none;`

`visibility: hidden;`

`opacity: 0;`

`position: absolute;  
top: -9999px;  
left: -9999px;`

---

Visually hidden?



---

Box model generated?



---

Affects layout?



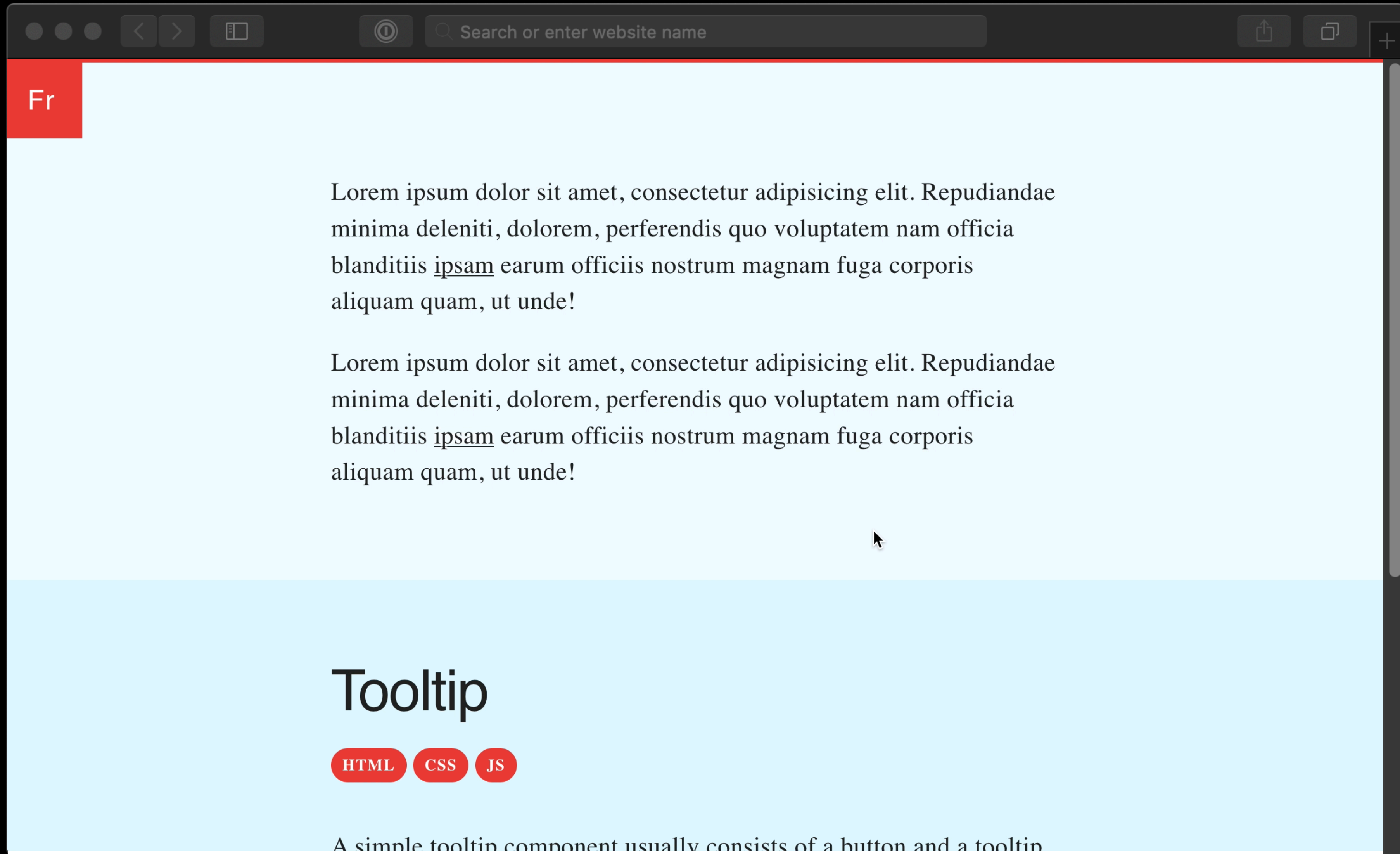
---

Read by assistive technologies?



```
display: none; visibility: hidden; opacity: 0; position: absolute; top: -9999px; left: -9999px;
```

Visually hidden?	✓	✓	✓	✓	✓
Box model generated?	✗	✓	✓	✓	✓
Affects layout?	✗	✓	✓	✗	✗
Read by assistive technologies?	✗	✗	✓	✓	✓



```
display: none;
```

```
visibility: hidden;
```

```
opacity: 0;
```

```
position: absolute;  
top: -9999px;  
left: -9999px;
```

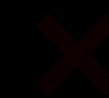
**Visually hidden?**



Box model generated?



Affects layout?



**Read by assistive technologies?**





```
display: none; visibility: hidden; opacity: 0; position: absolute; top: -9999px; left: -9999px;
```

Visually hidden?



Box model generated?



Affects layout?



Read by assistive technologies?



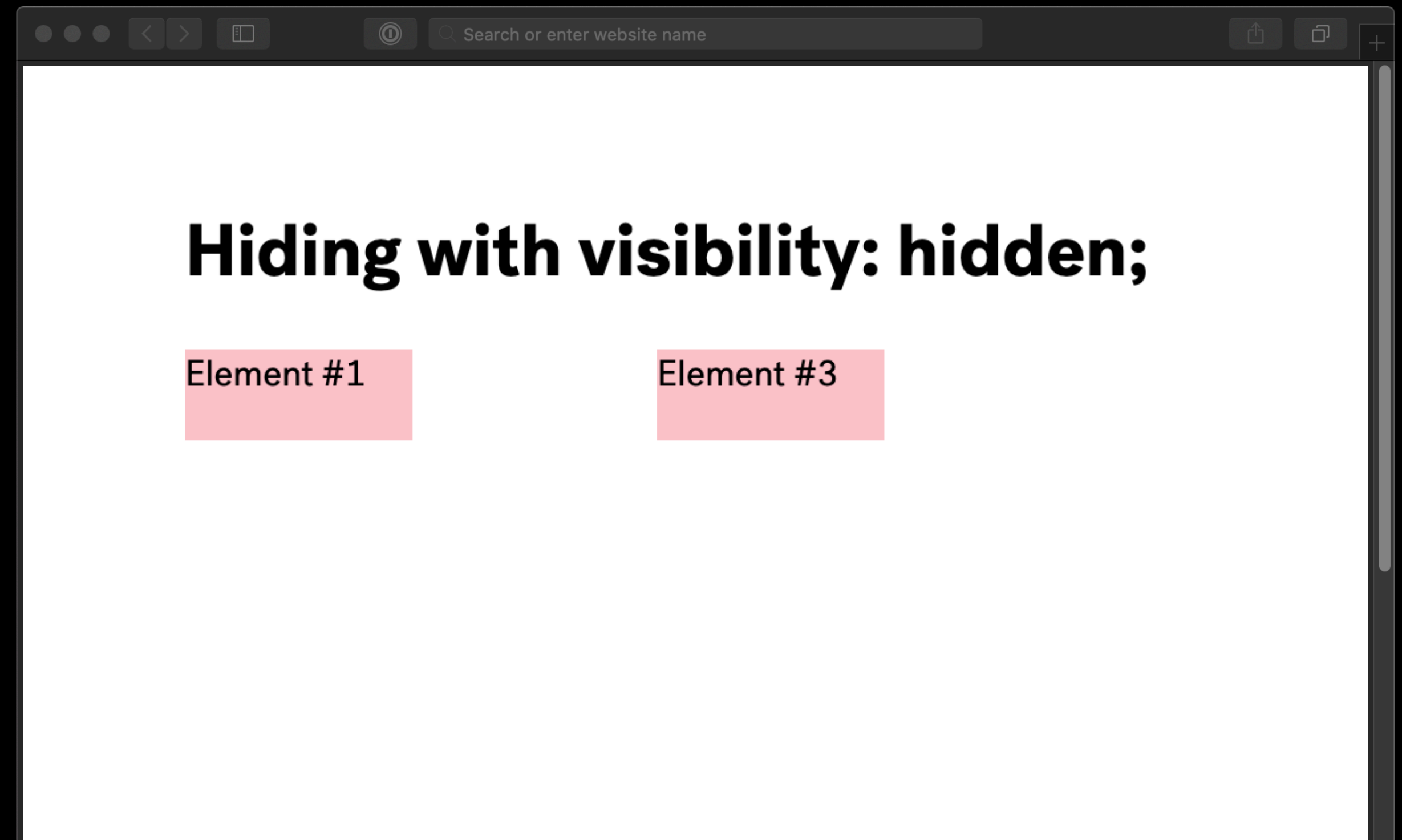
```
<div>Element #1</div>
```


```
<div style="visibility:hidden;">
```

```
  Element #2
```

```
</div>
```

```
<div>Element #3</div>
```



A young boy with dark hair and glasses is looking down at a wooden rocking horse. He is wearing a dark, patterned shirt. The room has a patterned rug and a bulletin board with various papers and photos pinned to it. The lighting is warm and somewhat dim.

**My body's gone!**

Don't use `visibility: hidden;`

```
display: none;    visibility: hidden;    opacity: 0;    position: absolute;    top: -9999px;    left: -9999px;
```

Visually hidden?



Box model generated?



Affects layout?



Read by assistive technologies?



```
display: none; visibility: hidden; opacity: 0;
```

```
position: absolute;  
top: -9999px;  
left: -9999px;
```

Visually hidden?



Box model generated?



Affects layout?



Read by assistive technologies?





**Source Order**



**Visual Order**

# Source order

```
<body>  
  <footer>...</footer>  
  <nav>...</nav>  
  <header>...</header>  
  <main>...</main>  
</body>
```

# Visual order

```
body { display: flex }  
header { order: 1 }  
nav { order: 2 }  
main { order: 3 }  
footer { order: 4 }
```



“With this power comes **great responsibility**”

— Rachel Andrew

<https://rachelandrew.co.uk/archives/2015/07/28/modern-css-layout-power-and-responsibility/>

#3

**Don't** write CSS that undoes the  
default accessible styles

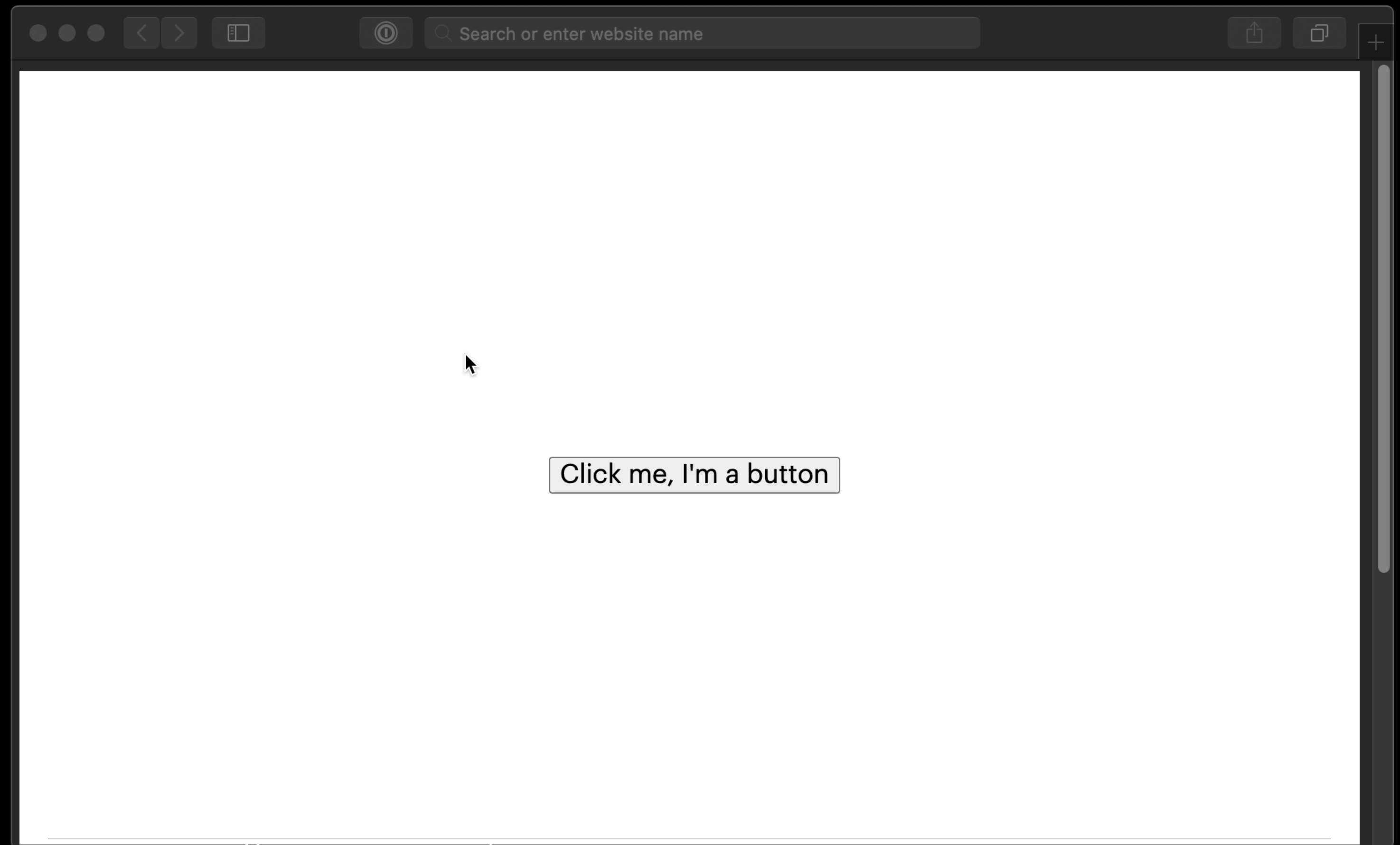


**Browser CSS**

✓ Text & element sizes

✓ Colours & contrast

✓ Hover, focus, & active states

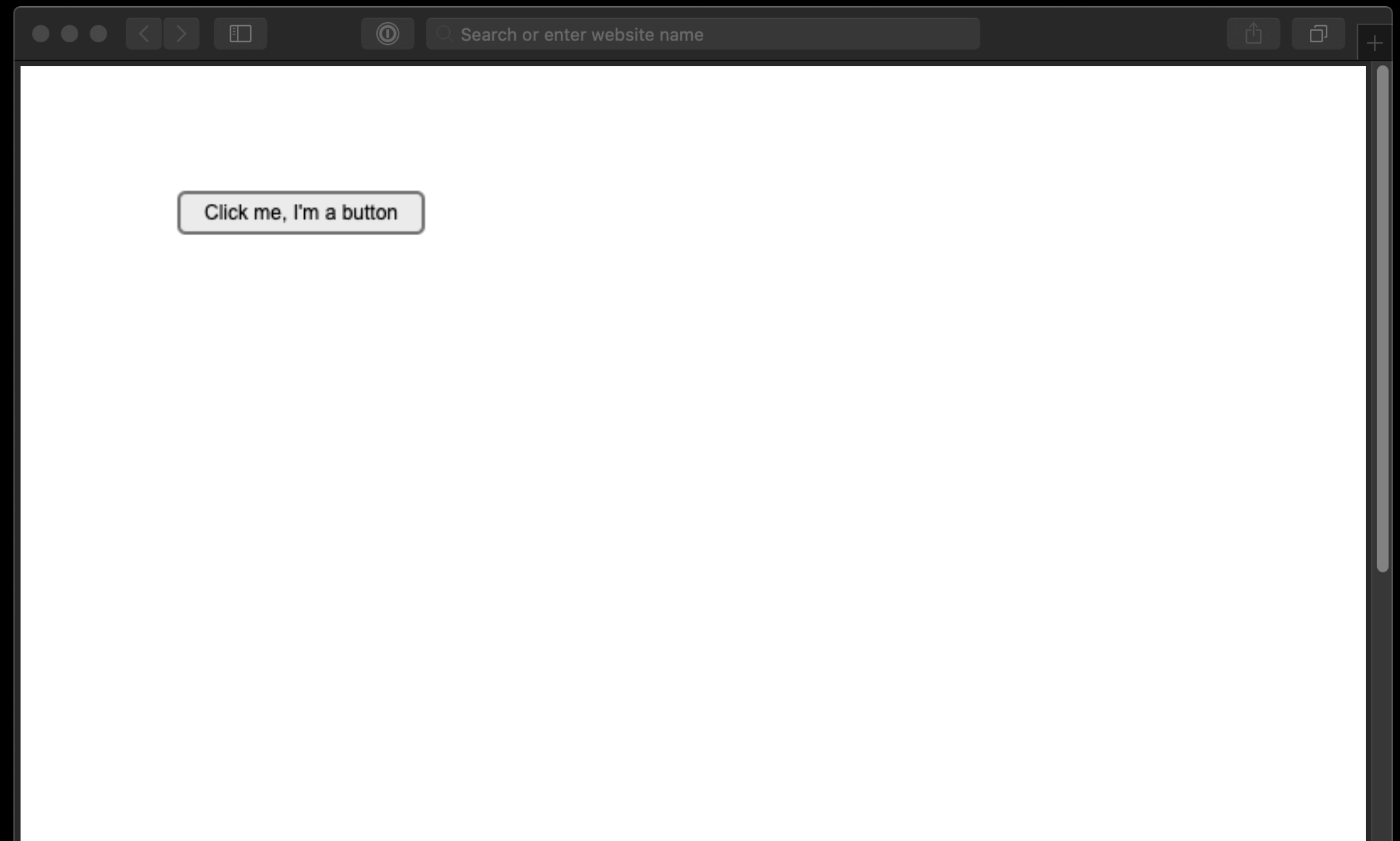


An illustration of a man in a dark suit with a yellow tie, standing on a dark blue globe. He is holding a telescope to his eye and looking towards the right. To his right is a large, stylized eye graphic with a dark blue iris and a light pinkish-white sclera. The background is a solid blue color with a few white clouds.

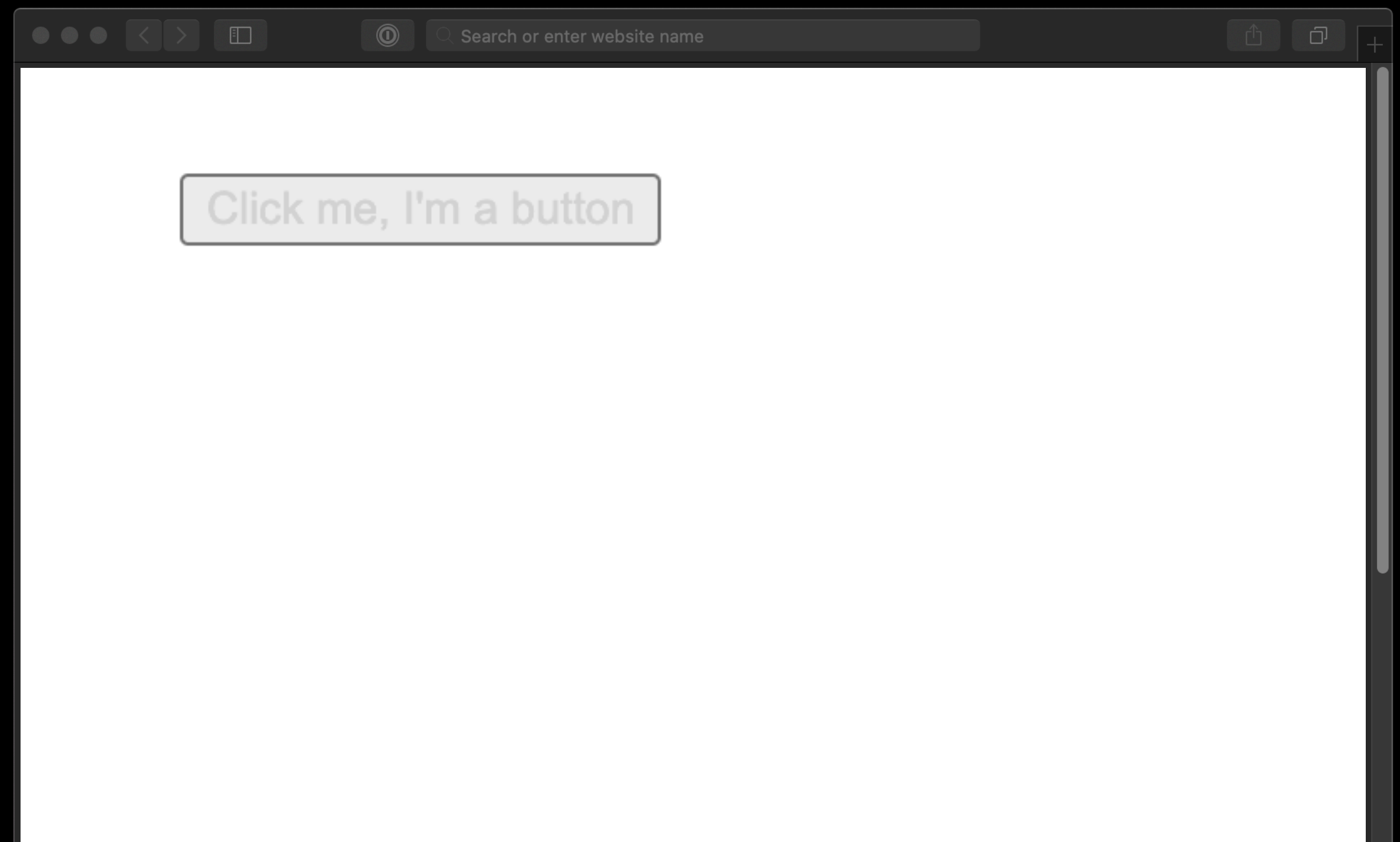
**My CSS**

**Browser CSS**

```
button {  
  font-size: 6px;  
}
```



```
button {  
    color: lightgray;  
}
```



hover

Button ↓

active

Button ↓

focus

Button ↓



## Recap

The `:hover`, `:focus`, and `:active` pseudo-classes allow us to style elements in response to how a user interacts with it. Depending on the device being used, these pseudo-classes become active at different points (or not at all).

	<code>:hover</code>	<code>:focus</code>	<code>:active</code>
Mouse 🖱️	Yes	Yes	Yes
Keyboard ⌨️	No	Yes	Yes
Touchscreen 📱	Yes (on click)	Yes* (on click)	Yes* (on click)

\* Will not apply on mobile (iOS) Safari

`:active then :focus then :hover` ❌

`:hover then :focus then :active` ✅

```
button:active { background-color: green; }
```

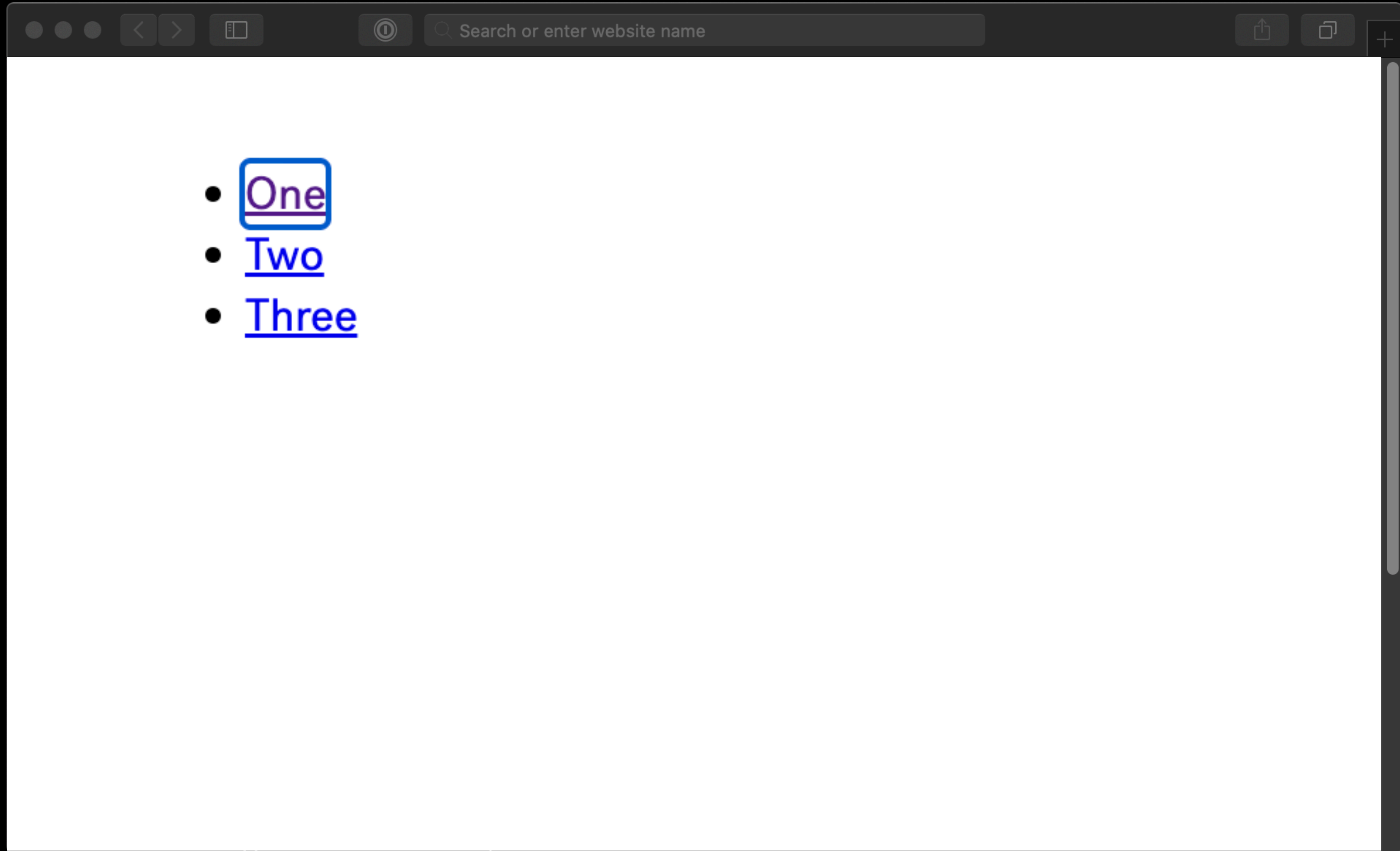
```
button:hover { background-color: red; }
```

```
button:focus { background-color: blue; }
```

```
button:focus { background-color: blue; }
```

```
button:hover { background-color: red; }
```

```
button:active { background-color: green; }
```



- One
- Two
- Three



```
:focus {  
  outline: none;  
}
```



```
:focus {  
  outline: none;  
  /* other focus styles */  
}
```

```
button:focus:not(:focus-visible) {  
  outline: none;  
}
```

```
button:focus-visible {  
  background-color: darksalmon;  
}
```



Write custom CSS **cautiously**



#4

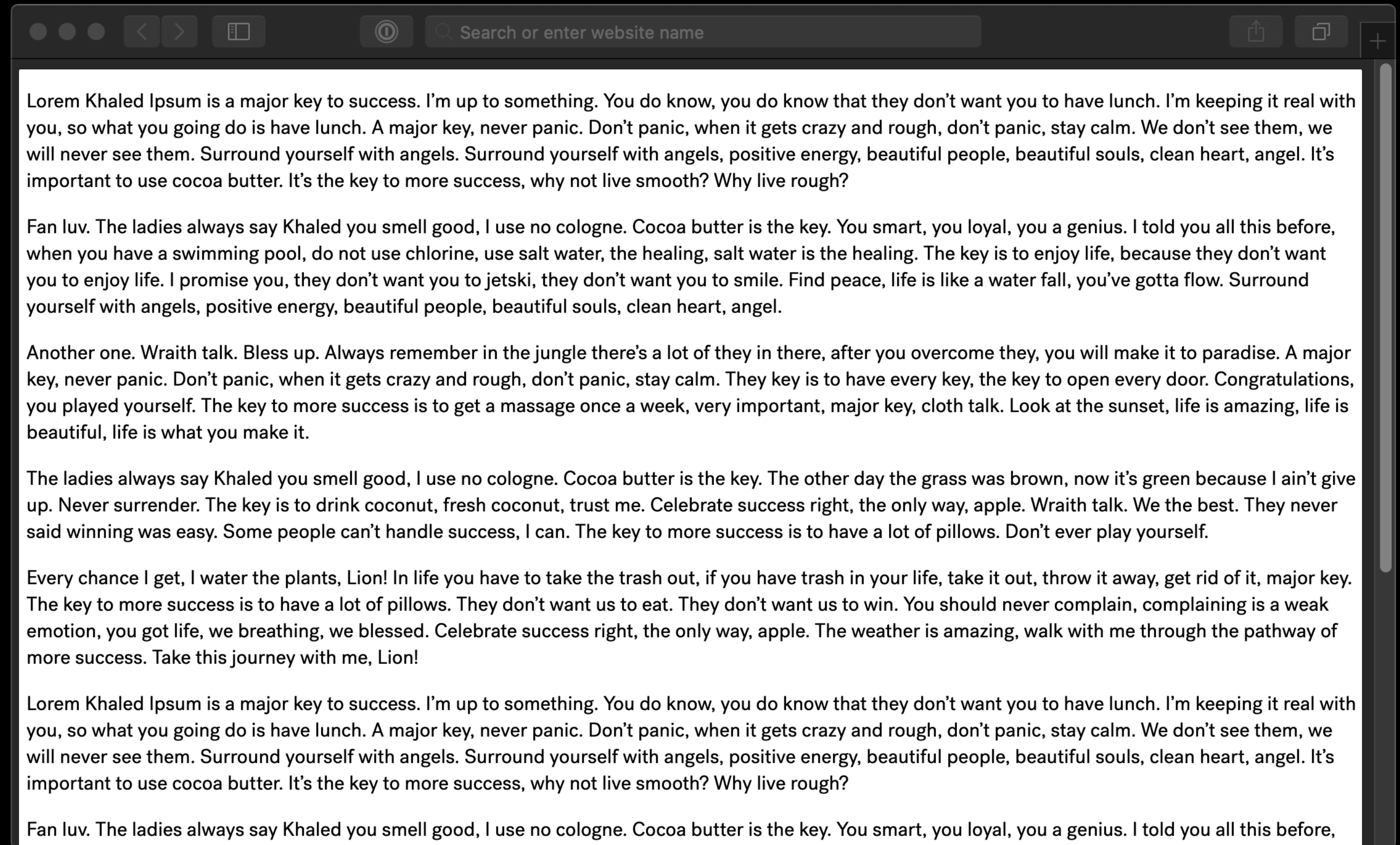
**Do** use CSS to improve on the default accessible styles



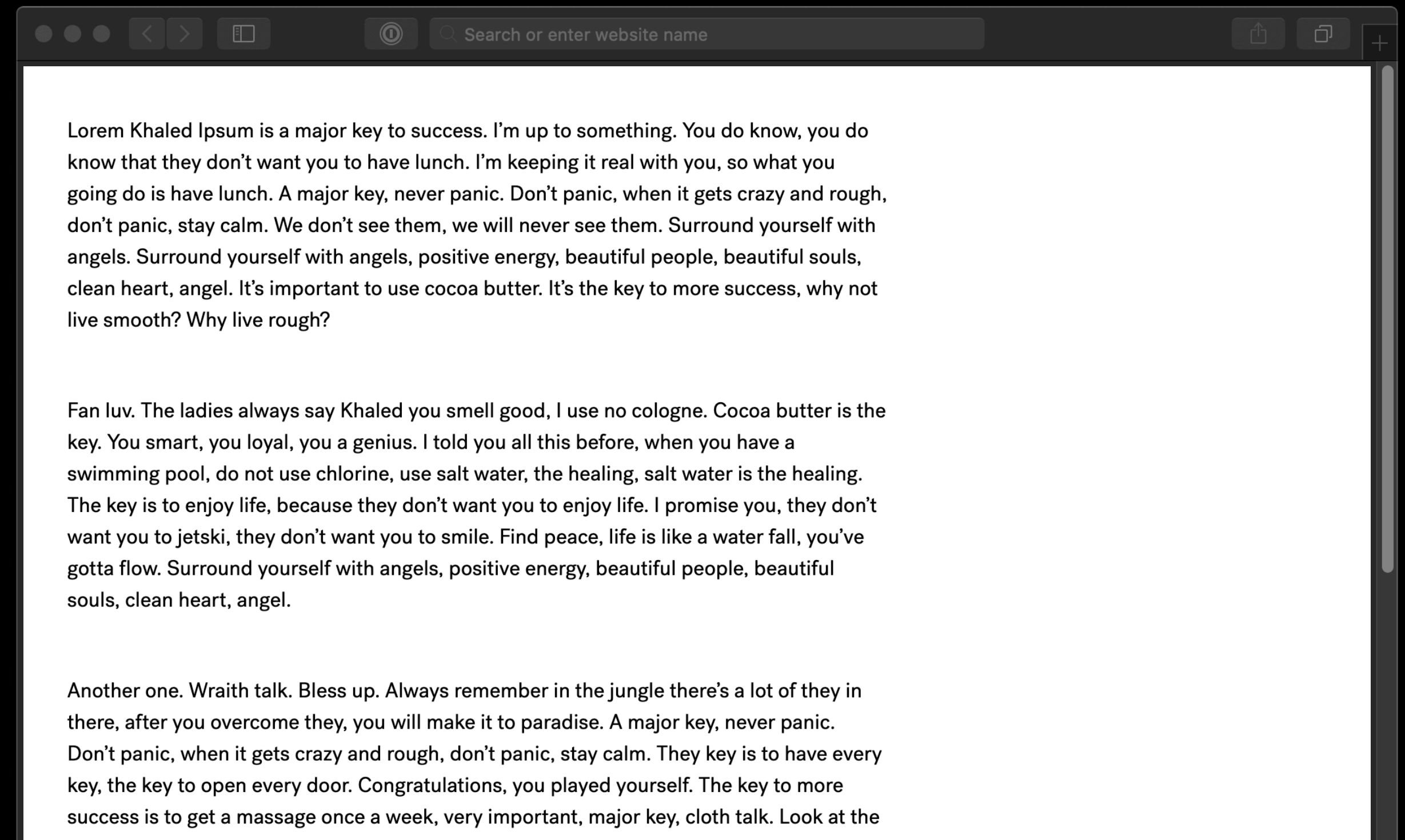
# Guideline 1.4.8 — Visual presentation

1. Text blocks should be **no wider than 80 characters**, for maximum readability.
2. Line height should be at least **1.5 times the text size** within paragraphs, and at least **2.25 times the text size** between paragraphs.

# The default browser styling **doesn't** meet these requirements



```
p {  
  
  line-height: 1.5;  
  
  padding: 2.25rem;  
  
  max-width: 80ch;  
  
}
```



## Guideline 2.4.1 — Bypass blocks

A mechanism should be provided that **allows the user to skip straight to the main content** or functionality available on the page, past the repeated features (such as the company logo or navigation).

# The Legend of Zelda: Tears of the Kingdom

28 languages

Contents [hide]

(Top)

- Gameplay
- Plot
- Development
- Reception
  - Sales
  - Accolades
- Notes
- References

Article Talk

Read View source View history Tools

From Wikipedia, the free encyclopedia

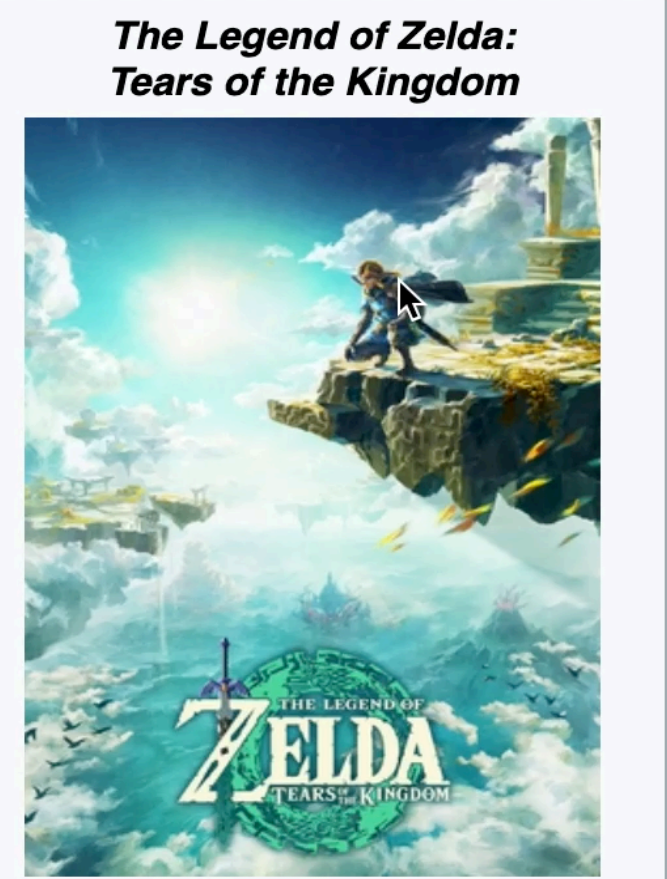
***The Legend of Zelda: Tears of the Kingdom***<sup>[b]</sup> is a 2023 [action-adventure game](#) developed and published by [Nintendo](#) for the [Nintendo Switch](#). The sequel to *The Legend of Zelda: Breath of the Wild* (2017), *Tears of the Kingdom* retains aspects including the [open world](#) of [Hyrule](#), which has been expanded to allow for more vertical exploration. The player controls [Link](#), who must help [Princess Zelda](#) to stop [Ganondorf](#) from destroying Hyrule.

*Tears of the Kingdom* was conceived after ideas for *Breath of the Wild* [downloadable content](#) (DLC) had exceeded its scope. Its development was led by Nintendo's [Entertainment Planning & Development](#) (EPD) division, with *Breath of the Wild* director [Hidemaro Fujibayashi](#) and producer [Eiji Aonuma](#) reprising their roles. A teaser was shown at [E3 2019](#) with a full reveal at [E3 2021](#). *Tears of the Kingdom* was initially planned for release in 2022 before being delayed to May 2023. It received acclaim for its improvements, expanded open world, and features encouraging exploration and experimentation. It sold more than 10 million copies in its first three days of release.

## Gameplay

*Tears of the Kingdom* retains the open-world action-adventure gameplay of the previous *Zelda* game, *Breath of the Wild* (2017). As [Link](#), players explore [Hyrule](#) and two new areas, the Sky Islands and the Depths, to find weapons, resources, and complete quests. Link can explore on foot or by climbing, horseriding and using paragliders.<sup>[1]</sup>

New to *Tears of the Kingdom* are the Zonai devices, which the player can use for combat, propulsion, exploration, and more. The previous game's runes are replaced with five new powers: Ultrahand, Fuse, Ascend, Recall, and Autobuild. Ultrahand allows the player to pick up and move different objects, and attach different objects together. This can be used with the Zonai devices to create different vehicles or other constructs. Fuse allows Link to combine materials, equipment, or certain objects in the world to a shield or a weapon to increase its attributes and durability. For example, fusing an explosive object to an arrow will cause the arrow to explode on impact. Autobuild instantly recreates a device crafted with Ultrahand, automatically using nearby devices and objects if available, or if parts are missing, creating



<b>Developer(s)</b>	Nintendo EPD <sup>[a]</sup>
<b>Publisher(s)</b>	Nintendo
<b>Director(s)</b>	Hidemaro Fujibayashi
<b>Producer(s)</b>	Eiji Aonuma
<b>Designer(s)</b>	Mari Shirakawa Naoki Mori Akihito Toda
<b>Programmer(s)</b>	Takahiro Okuda
<b>Artist(s)</b>	Satoru Takizawa
<b>Composer(s)</b>	Manaka Kataoka Maasa Miyoshi

## Guideline 2.4.1 — Bypass blocks

A mechanism should be provided that allows the user to skip straight to the main content or functionality available on the page, past the repeated features (such as the company logo or navigation).

...

If a proper structure of headings and semantic containers is provided to navigate with (for example <section>, <aside>, etc.), **then an added "skip link" is not needed.**

#5

**Do** adapt CSS to device capabilities

Keeps your  
dishes sparkling!

CSS





```
.element {
```

```
    color: #000;
```

```
    color: var(-text-color);
```

```
}
```

```
.element {
```

```
  color: #000;
```

```
  color: var(-text-color);
```

```
}
```

```
.element {
```

```
  color: #000;
```

```
  color: var(--text-color);
```

```
}
```

```
@supports ( declaration ) {  
    /* Feature-based CSS here */  
}
```

# Legacy code

```
main {  
  display: table;  
}
```

# Modern code

```
@supports (display: grid) {  
  main {  
    display: grid;  
  }  
  /* more layout code */  
}
```

# Legacy code

```
main {  
    display: table;  
}
```

# Modern code

```
@supports (display: grid) {  
    main {  
        display: grid;  
    }  
    /* more layout code */  
}
```

#6

**Do** adapt CSS to user preferences

# User Stylesheets in CSS

## Introduction

User stylesheets are an exciting feature of **Cascading Style Sheets (CSS)**. In **CSS**, the presentation of a document is controlled by the combination of user and author style preferences. This mechanism is **needed [text only]** to allow CSS to describe fully (and then extend) the current behavior of browsers. Early implementations of **CSS** did not support user stylesheets. However, newer browsers, such as **MS Internet Explorer 4.0+**, **Opera 3.50+**, and hopefully Netscape Navigator 5.0 (based on the **Mozilla** project) all support user stylesheets.

The interaction of user and author stylesheets requires stylesheets to be well behaved in certain ways. The main purpose of this document is to explain what a good user stylesheet is. I will also try to comment on what good author stylesheets are. Some of this discussion involves discussion of specific properties, while some is more general. Finally, I will address bugs in browsers' handling of user stylesheets.



## Chromium Code Reviews

### Issue [64843004](#): Get rid of user-level styles. (Closed)

**Created:**

7 years, 1 month ago by ojan

**Modified:**

7 years, 1 month ago

**Reviewers:**

[darin](#) (slow to review)

**CC:**

chromium-reviews, extensions-reviews\_chromium.org, jam, joi+watch-content\_chromium.org, darin-cc\_chromium.org, chromium-apps-reviews\_chromium.org, jochen+watch\_chromium.org

**Base URL:**

<svn://svn.chromium.org/chrome/trunk/src>

**Visibility:**

Public.

**Description**

Get rid of user-level styles.

-The Apps codepath for this is just using the wrong thing. It should be using author-level styles like extensions do.  
-The user-stylesheet feature requires the user to put a CSS stylesheet in the right location in their user-data-dir. Extensions are a much better way of doing this.

This is in preparation for simplifying the Blink style resolution code by removing the concept of user styles.

Committed: <https://src.chromium.org/viewvc/chrome?view=rev&revision=234007>

**Patch Set 1****Patch Set 2 : merge to ToT**

Created: 7 years, 1 month ago

Unified diffs	Side-by-side diffs	Stats (+1 line, -488 lines)
▶ M <a href="#">chrome/browser/chrome_content_browser_client.cc</a>	<a href="#">View</a>	2 chunks +0 lines, -17 lines 0 comments
M <a href="#">chrome/browser/profiles/chrome_browser_main_extra_parts_profiles.cc</a>	<a href="#">View</a>	2 chunks +0 lines, -4 lines 0 comments
M <a href="#">chrome/browser/profiles/profile_impl.cc</a>	<a href="#">View</a>	1 chunk +0 lines, -1 line 0 comments
M <a href="#">chrome/browser/ui/prefs/prefs_tab_helper.cc</a>	<a href="#">View</a>	2 chunks +0 lines, -11 lines 0 comments
D <a href="#">chrome/browser/user_style_sheet_watcher.h</a>	<a href="#">View</a>	1 chunk +0 lines, -68 lines 0 comments
D <a href="#">chrome/browser/user_style_sheet_watcher.cc</a>	<a href="#">View</a>	1 chunk +0 lines, -215 lines 0 comments
D <a href="#">chrome/browser/user_style_sheet_watcher_factory.h</a>	<a href="#">View</a>	1 chunk +0 lines, -39 lines 0 comments
D <a href="#">chrome/browser/user_style_sheet_watcher_factory.cc</a>	<a href="#">View</a>	1 chunk +0 lines, -51 lines 0 comments

“All users, including users with disabilities,  
[should] have equal **control over the**  
**environment they use** to access the web”

```
@media ( prefers-* ) {
```

```
    /* Preference-based CSS here */
```

```
}
```

# Adapt to **motion** preferences

```
.element { animation: bouncing 1.5s linear infinite alternate; }
```

```
@media ( prefers-reduced-motion: reduce ) {
```

```
  .element { animation: fade 0.5s ease-in both; }
```

```
}
```

System Settings window with a dark theme. The left sidebar shows various settings categories, with 'Accessibility' highlighted in blue. The main pane is titled 'Display' and contains several settings:

- Display**
  - Invert colours:
  - Invert colours mode:  Smart  Classic
  - Reduce motion:  (indicated by a purple arrow)
  - Dim flashing lights:   
Video content that depicts repeated flashing or strobing lights will be automatically dimmed.
  - Increase contrast:
  - Reduce transparency:
  - Differentiate without colour:
  - Show window title icons:
  - Show toolbar button shapes:
  - Menu bar size:  Default  Large
  - Display contrast: Slider from Normal to Maximum
- Pointer**
  - Shake mouse pointer to locate:

Adapt to **motion** preferences

---

`prefers-reduced-motion`

Adapt to **data** preferences

---

`prefers-reduced-data`

Adapt to **colour** preferences

---

`prefers-color-scheme`

Adapt to **contrast** preferences

---

`prefers-contrast`

Adapt to **transparency** preferences

`prefers-reduced-transparency`

Giving users **control**

# CSS & Accessibility

1. **Don't** use CSS to convey meaning or content
2. **Don't** use CSS to change the semantics of HTML
3. **Don't** write CSS that undoes the default accessible styles

1. **Do** use CSS to improve on the default accessible styles
2. **Do** adapt CSS to device capabilities
3. **Do** adapt CSS to user preferences



**Part Two**

# **JavaScript & Accessibility**

JavaScript is used to make web pages  
**more interactive**

JavaScript is used to make web pages

***\*more\** interactive**

**<a>**

**<details>**

**<form>**

**<summary>**

**<dialog>**

**<input>**

**<select>**

**<option>**

**<textarea>**

**<progress>**



#1

**Don't** use JavaScript for functionality

HTML provides

✓ Triggered by mouse, enter key, and space bar

✓ Focusable via the keyboard and other input devices

✓ Accessible name and state provided to assistive tech



```
<button>
```

```
  Do something
```

```
</button>
```



```
<div>
```

```
  Do something
```

```
</div>
```

# Extra work 🥵

```
<div tabindex="0"  
  role="button"  
  onKeyPress="handleBtnKeyPress(e)"  
  onClick="doSomething(e)">  
  Do something  
</div>
```

# Extra work 🥵

```
<div tabIndex="0"  
  role="button"  
  onKeyPress="handleBtnKeyPress(e)"  
  onClick="doSomething(e)">  
  Do something  
</div>
```

# Extra work 🥵

```
<div tabindex="0"  
  role="button"  
  onKeyPress="handleBtnKeyPress(e)"  
  onClick="doSomething(e)">  
  Do something  
</div>
```

# More extra work 🥲

```
<div tabIndex="0"  
  role="button"  
  onKeyPress="handleBtnKeyPress(e)"  
  onClick="doSomething(e)">  
  
  Do something  
  
</div>
```

```
function handleBtnKeyPress(e) {  
  if ( e.keyCode === 32 ||  
      e.keyCode === 13) {  
    doSomething(e);  
  }  
}
```

# Even more extra work 🥲

```
<div tabIndex="0"  
  role="button"  
  onKeyPress="handleBtnKeyPress(e)"  
  onClick="doSomething(e)"  
  aria-pressed="false">  
  Toggle  
</div>
```





```
<a href="/about">About Us</a>
```



```
<button onClick="goToPage( '/about' )">
```

About Us

```
</button>
```



```
<button onClick="goToPage( '/about' )">
```

```
    About Us
```

```
</button>
```

```
function goToPage(url) {
```

```
    // Do some other things
```

```
    window.location.href = url;
```

```
}
```

Keep JavaScript enhancements **unobtrusive**

```
<a href="/about" onClick="goToPage(e, '/about')">
```

```
    About Us
```

```
</a>
```

```
function goToPage(e, url) {
```

```
    e.preventDefault();
```

```
    // Do some other things
```

```
    window.location.href = url;
```

```
}
```

```
<a href="/about" onClick="goToPage(event, '/about')">
```

```
    About Us
```

```
</a>
```

```
function goToPage(event, url) {
```

```
    event.preventDefault();
```

```
    // Do some other things
```

```
    window.location.href = url;
```

```
}
```

```
<a href="/about" onClick="goToPage(e, '/about')">
```

About Us

```
</a>
```

```
function goToPage(e, url) {
```

```
  e.preventDefault();
```

```
  // Do some other things
```

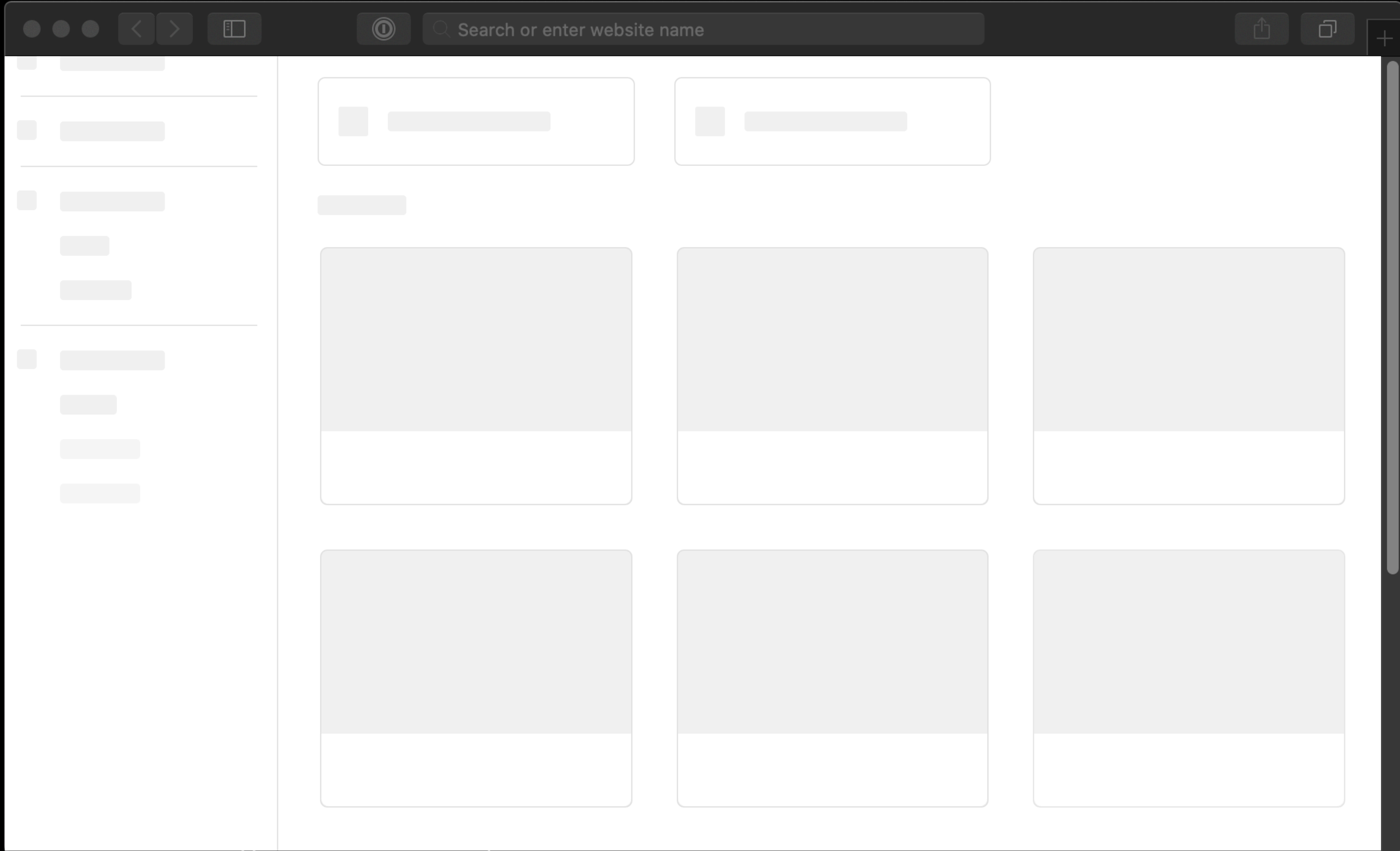
```
  window.location.href = url;
```

```
}
```

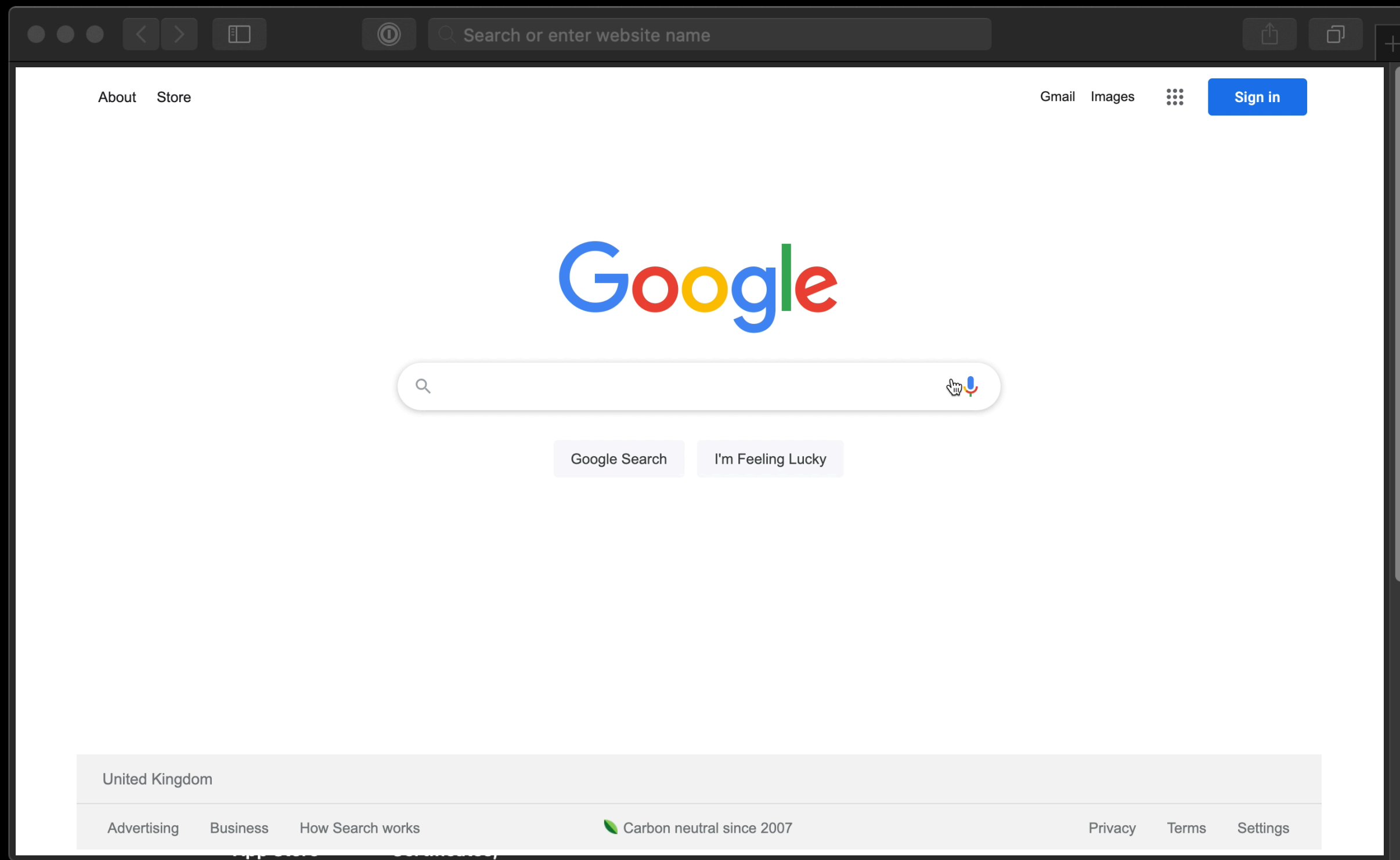
`event.preventDefault()` is the 

#2

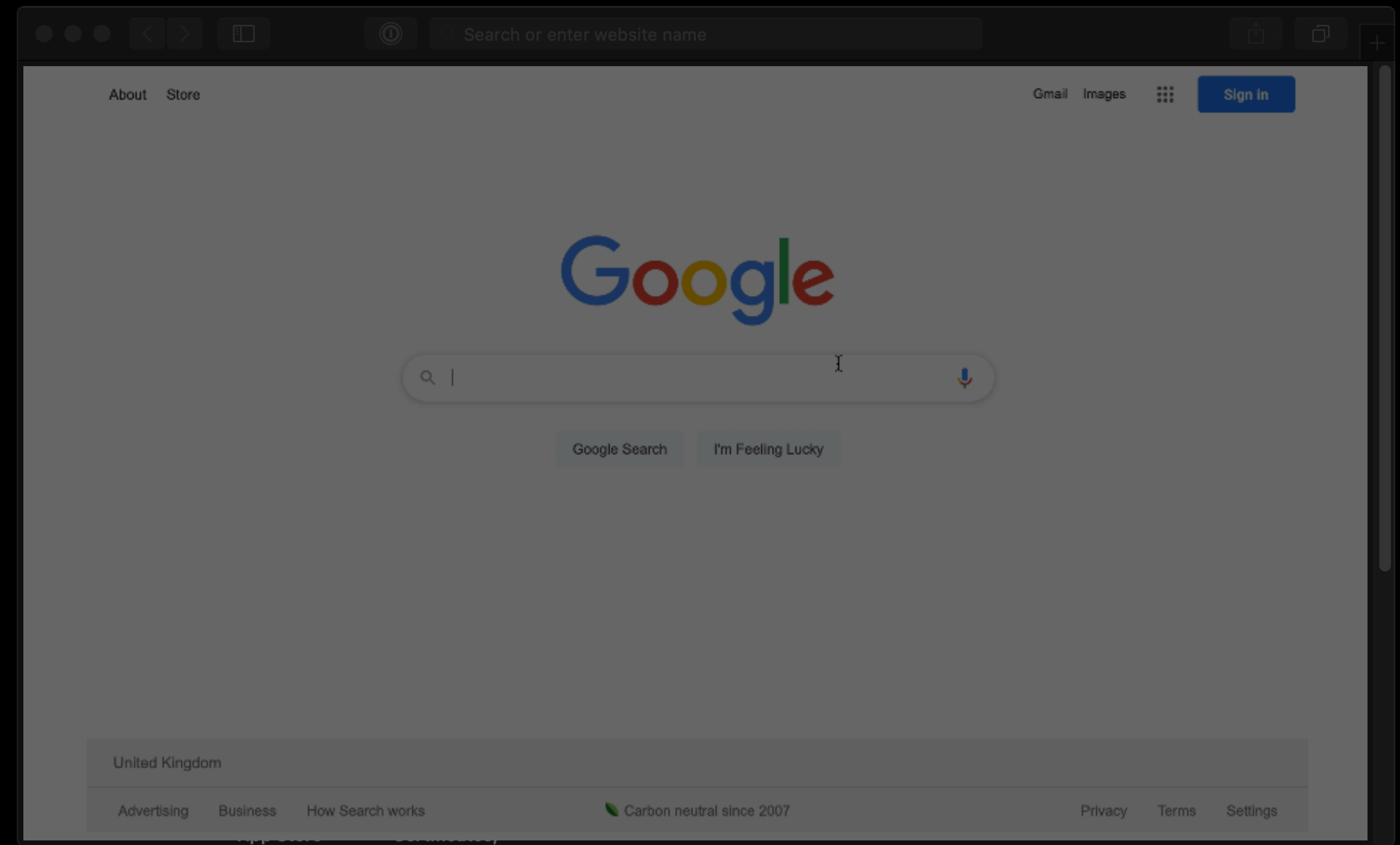
Where possible, **don't** require  
JavaScript for critical features



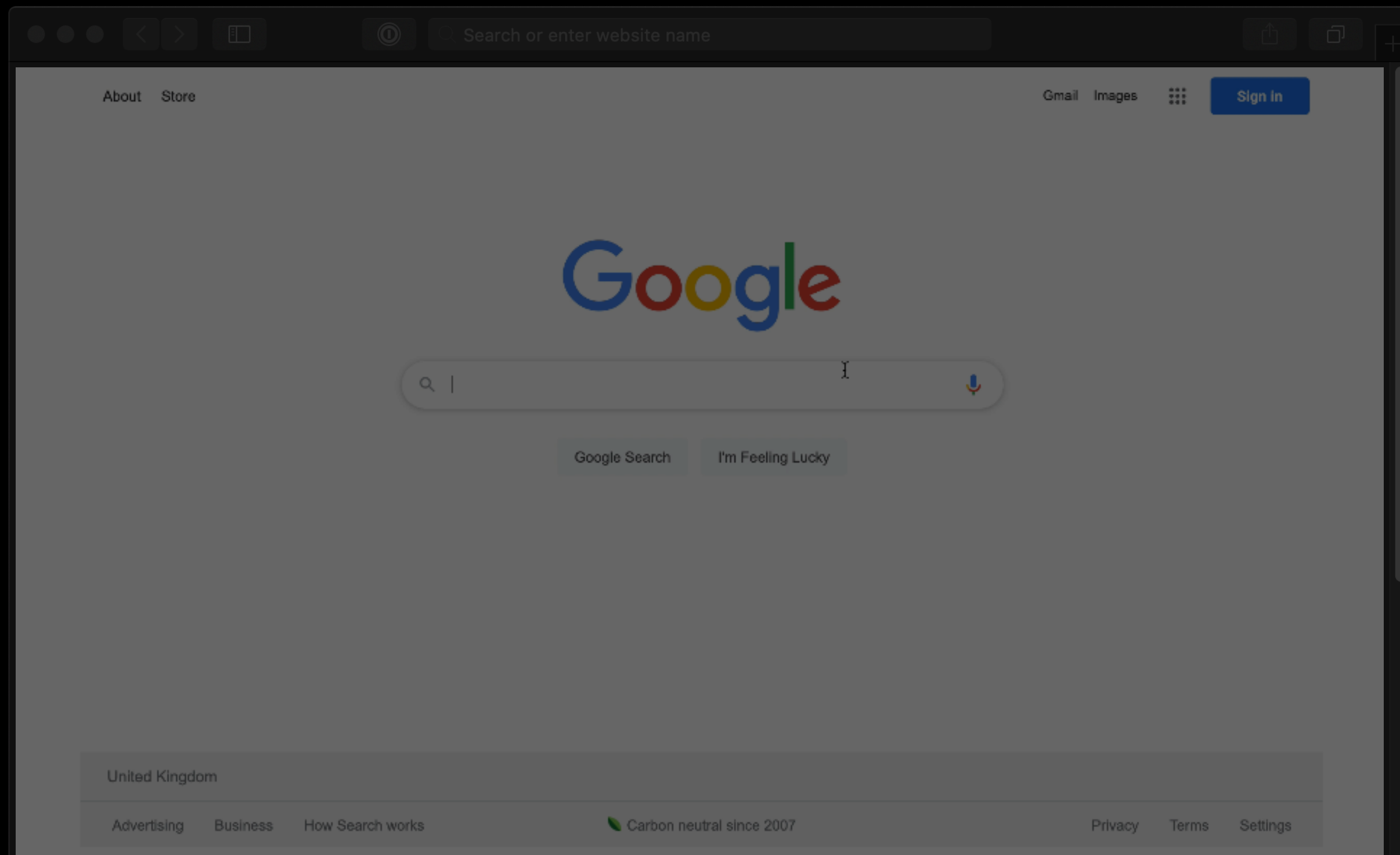




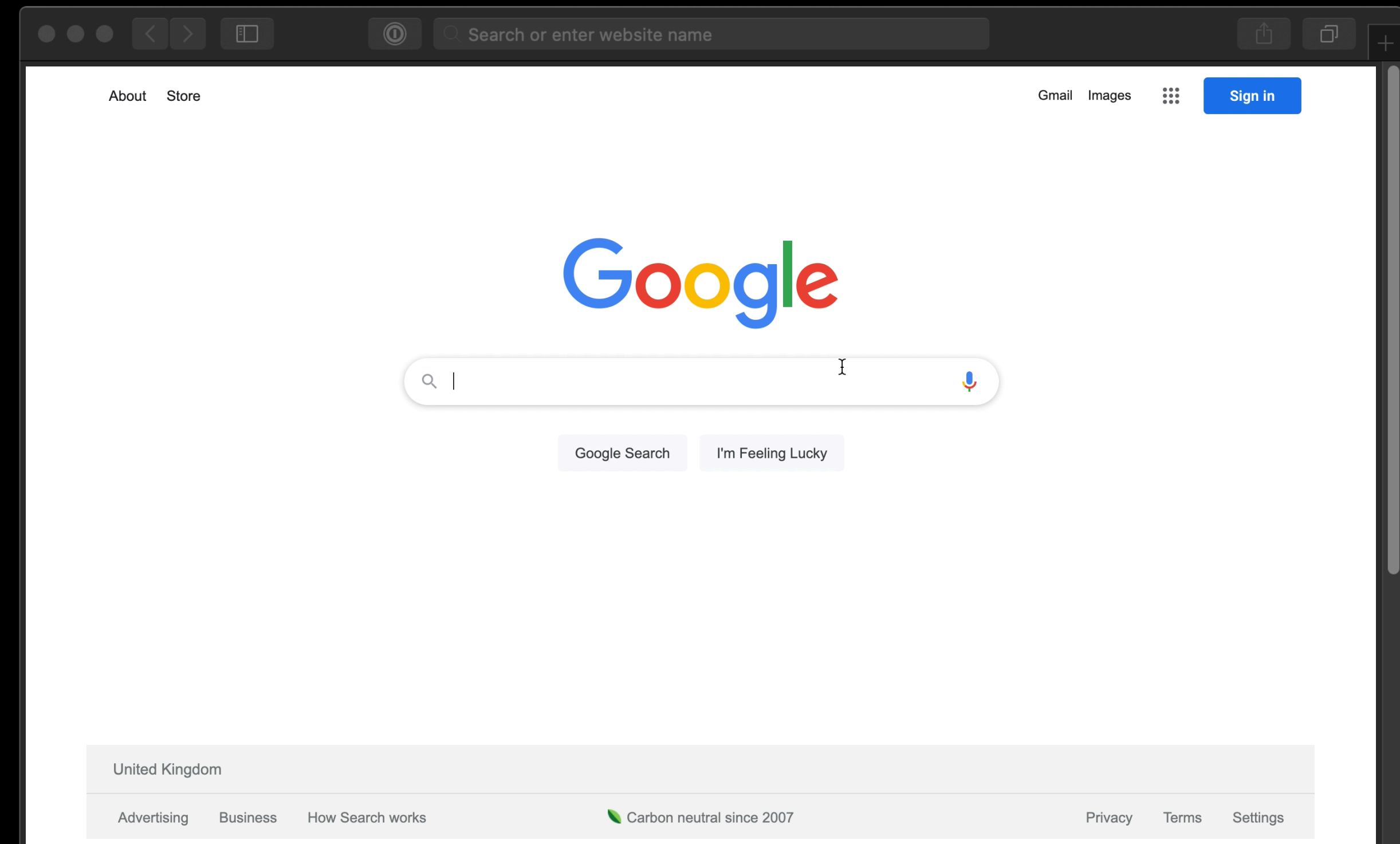
**Without JS**



**With JS**



**Without JS**



**With JS**

“While WCAG 1.0 from 1999 required that pages be functional and accessible with scripting disabled, **WCAG 2 and all other modern guidelines allow you to require JavaScript**”

<https://webaim.org/techniques/javascript/>

# Different groups have different access needs



**People with physical disabilities are more likely to **rely on JS****

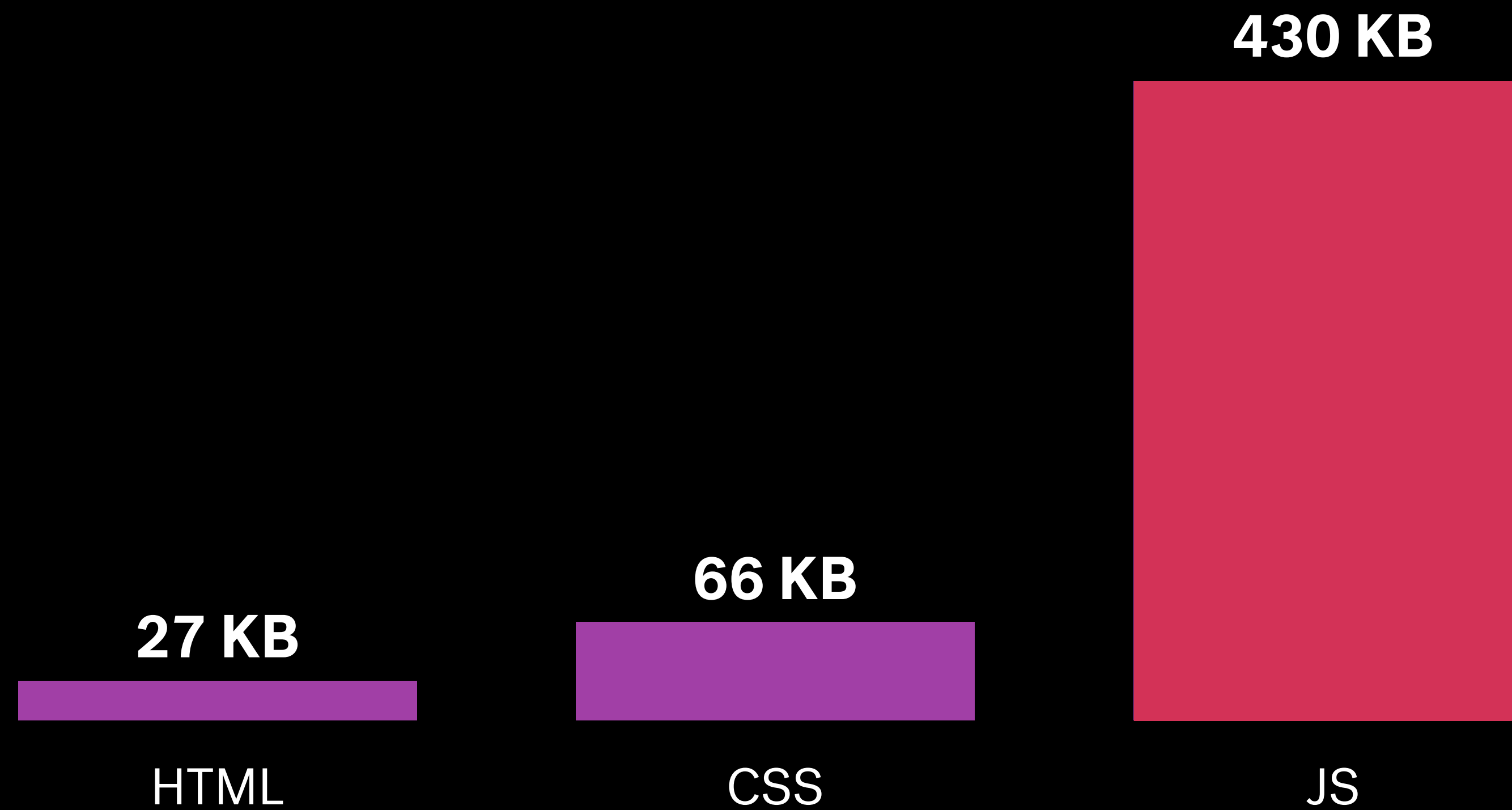
Only 0.07% of screen reader users have JavaScript disabled.

The global average is 1%.

**People with socio-economic  
restrictions are more likely to avoid JS**

Over 50% of the Sudanese mobile browsing is with Opera Mini

# Average File Sizes







**JavaScript-disabled**  
experience

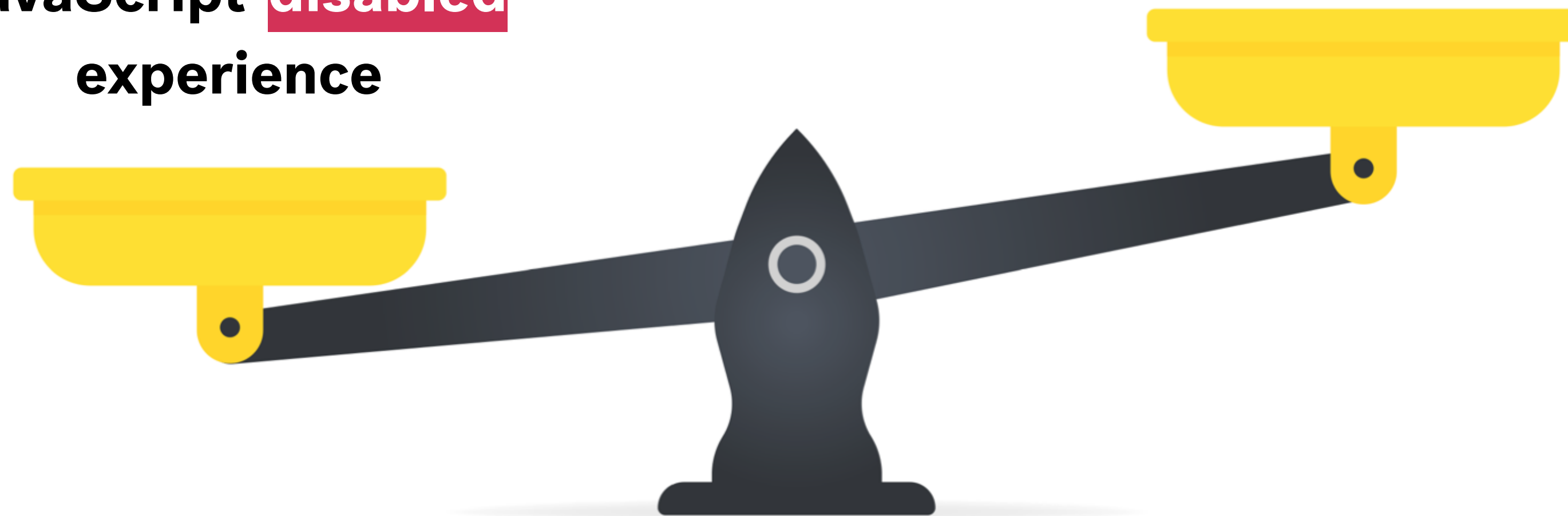


**JavaScript-enabled**  
experience



**JavaScript-disabled  
experience**

**JavaScript-enabled  
experience**



**JavaScript-disabled**  
experience

**JavaScript-enabled**  
experience



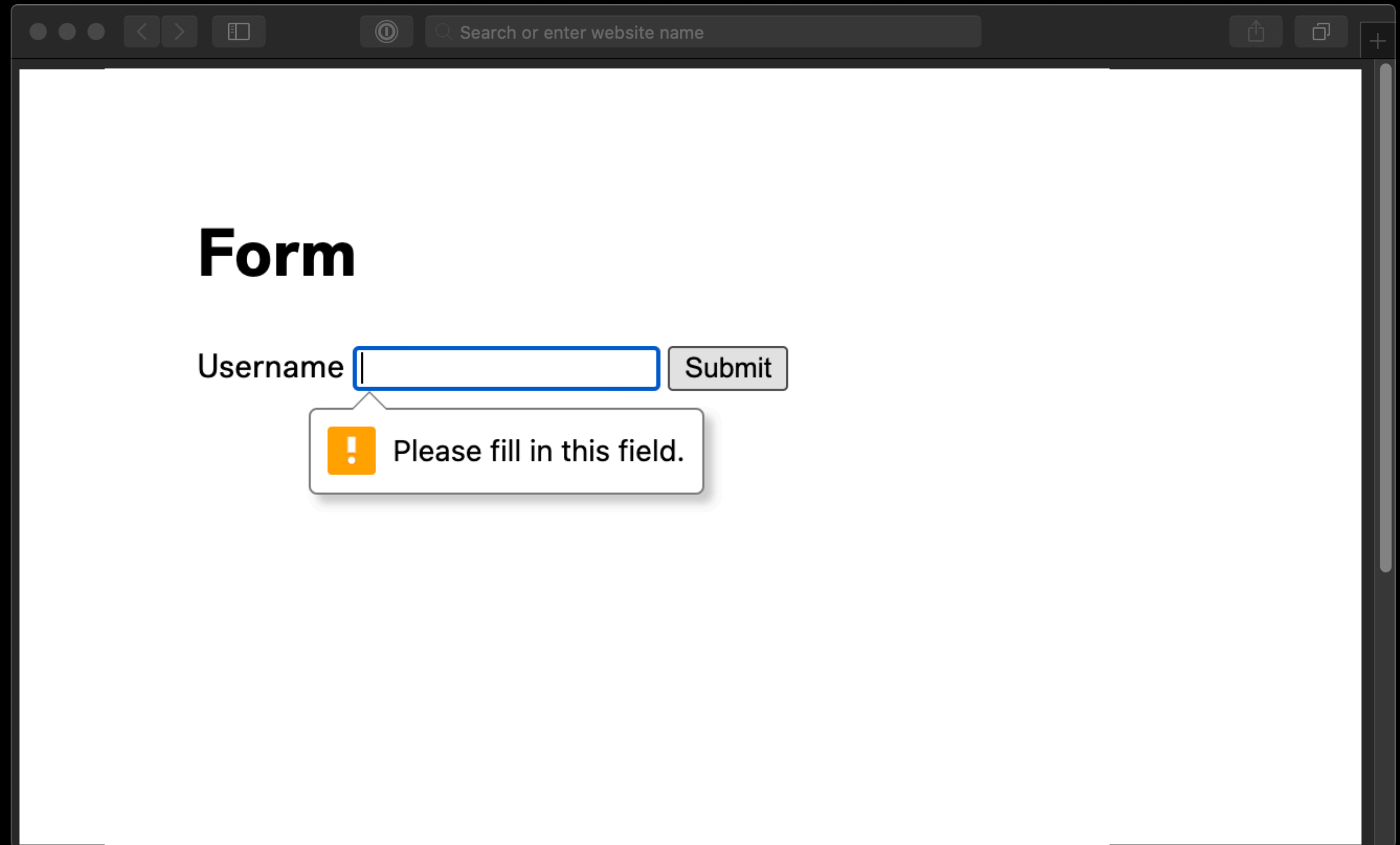
“Just because JavaScript is used on a page **does not mean that the page is inaccessible**. In many cases, JavaScript can be used to greatly improve accessibility and optimize the user experience.”

“Just because JavaScript is used on a page does not mean that the page is inaccessible. In many cases, **JavaScript can be used to greatly improve accessibility** and optimize the user experience.”

#3

**Do** use JavaScript to improve on the default accessible behaviour

**Guideline 3.3 — Help  
users avoid and  
correct mistakes** ✓



The image shows a browser window with a search bar at the top containing the text "Search or enter website name". Below the search bar, the word "Form" is displayed in a large, bold, black font. Underneath "Form", there is a label "Username" followed by an empty text input field. To the right of the input field is a "Submit" button. A white tooltip with a thin grey border and a drop shadow is positioned below the input field. The tooltip contains an orange square icon with a white exclamation mark, followed by the text "Please fill in this field.".

**Guideline 3.3.3 — When an error is detected and suggestions for correction are known, provide these to the user**



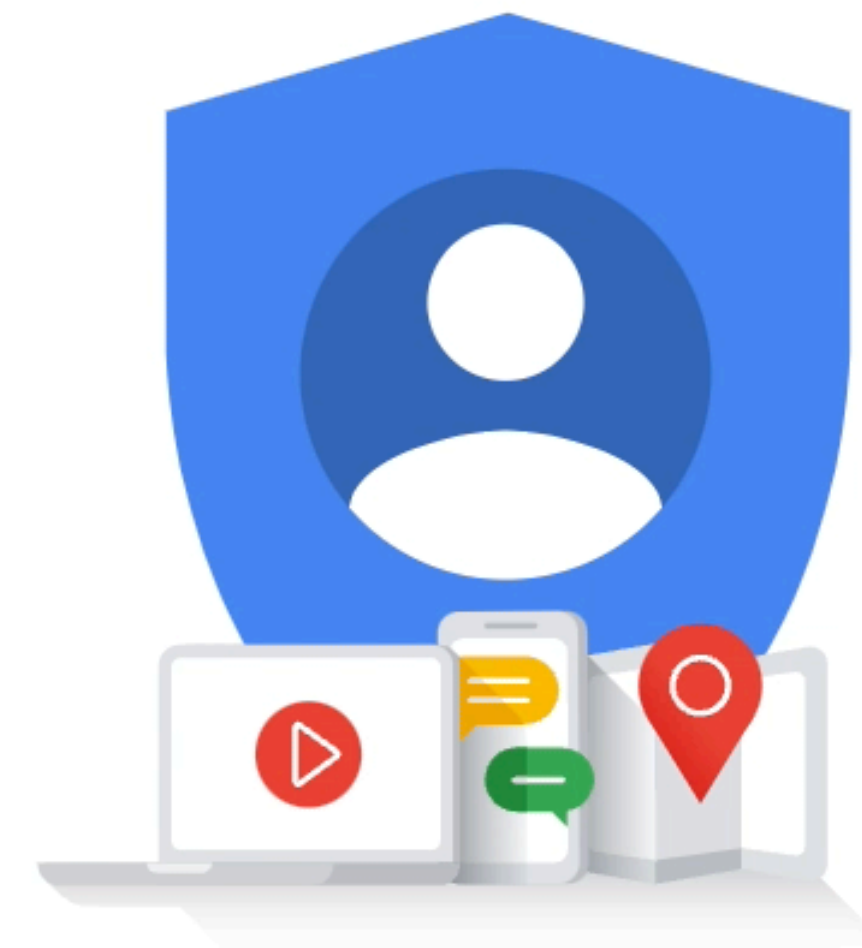


# Create your Google Account

to continue to Gmail

You can use letters, numbers & periods

Use 8 or more characters with a mix of letters, numbers &



Some HTML elements **aren't accessible**  
**enough**, yet 😞

# Problems with `<video>`

- ✗ Controls not focusable via keyboard
- ✗ Can't pause/play video using space key
- ✗ No arrow key support for scrubber

& more

“Modern browsers provide a default media player. **Most have limited functionality to support accessibility.**”

<https://www.w3.org/WAI/media/av/player/>

Search or enter website name

**DigitalA11Y**  
Your Accessibility Partner

Search DigitalA11Y

[Home](#) » Accessible HTML5 Media Players & Resources

**HTML · WEB ACCESSIBILITY**

# Accessible HTML5 Media Players & Resources

 Authored By : Raghavendra Satish Peri • Last Updated : January 20, 2022 • HTML, Web Accessibility

who access the internet, they have the skills and the tools to read content on the internet.

Sometimes, we **need** JavaScript

#4

**Do** use JavaScript to create  
components that don't exist





**<tooltip>**

**<dropdown>**

**<carousel>**

**<tab-group>**

**<accordion>**

**<toggle>**


**<card>**

**<loading>**

**<social-button>**

**<tab>**

Example [Open In CodePen](#)



**Travel to Southwest England and Paris**  
Sept. 14 to Sept. 24 or 27

# Guidelines related to carousels

1.3.1 — Information, structure, and relationships conveyed through presentation can be programmatically determined

2.1.1 — All functionality should be accessible using keyboard controls

2.2.2 — Controls should be provided to pause, stop, or hide moving content

4.1.2 — The name and role of user interface components (e.g. form inputs, buttons, links, etc.) should be programmatically determinable

# Guidelines related to carousels

**1.3.1** — Information, structure, and relationships conveyed through presentation can be programmatically determined

2.1.1 — All functionality should be accessible using keyboard controls

2.2.2 — Controls should be provided to pause, stop, or hide moving content

4.1.2 — The name and role of user interface components (e.g. form inputs, buttons, links, etc.) should be programmatically determinable

```
<ul id="slides">
```

```
  <li>/* Slide One */</li>
```

```
  <li>/* Slide Two */</li>
```

```
  <li>/* Slide Three */</li>
```

```
</ul>
```

# Guidelines related to carousels

1.3.1 — Information, structure, and relationships conveyed through presentation can be programmatically determined

**2.1.1** — All functionality should be accessible using keyboard controls

**2.2.2** — Controls should be provided to pause, stop, or hide moving content

4.1.2 — The name and role of user interface components (e.g. form inputs, buttons, links, etc.) should be programmatically determinable

```
<section>
  <div>
    <button>Toggle Play Slideshow</button>
    <button>Previous Slide</button>
    <button>Next Slide</button>
  </div>
  <ul id="slides">
    <li> ... </li>
  </ul>
</section>
```

```
<section>
  <div>
    <button>Toggle Play Slideshow</button>
    <button>Previous Slide</button>
    <button>Next Slide</button>
  </div>
  <ul id="slides">
    <li> ... </li>
  </ul>
</section>
```



```
<section>
  <div>
    <button>Toggle Play Slideshow</button>
    <button>Previous Slide</button>
    <button>Next Slide</button>
  </div>
  <ul id="slides">
    <li> ... </li>
  </ul>
</section>
```

# Guidelines related to carousels

- 1.3.1 — Information, structure, and relationships conveyed through presentation can be programmatically determined
- 2.1.1 — All functionality should be accessible using keyboard controls
- 2.2.2 — Controls should be provided to pause, stop, or hide moving content
- 4.1.2** — The name and role of user interface components (e.g. form inputs, buttons, links, etc.) should be programmatically determinable

```
<section aria-roledescription="carousel" aria-label="Slideshow">
  <div>
    <button>Toggle Play Slideshow</button>
    <button aria-controls="slides">Previous Slide</button>
    <button aria-controls="slides">Next Slide</button>
  </div>
  <ul id="slides">
    <li role="group" aria-roledescription="slide" aria-label="1 of 3"> ... </li>
  </ul>
</section>
```

Write custom components **cautiously**

Jump To Content (Option+0)

[WAI-ARIA Authoring Practices 1.2](#)

## Index of ARIA Design Pattern Examples

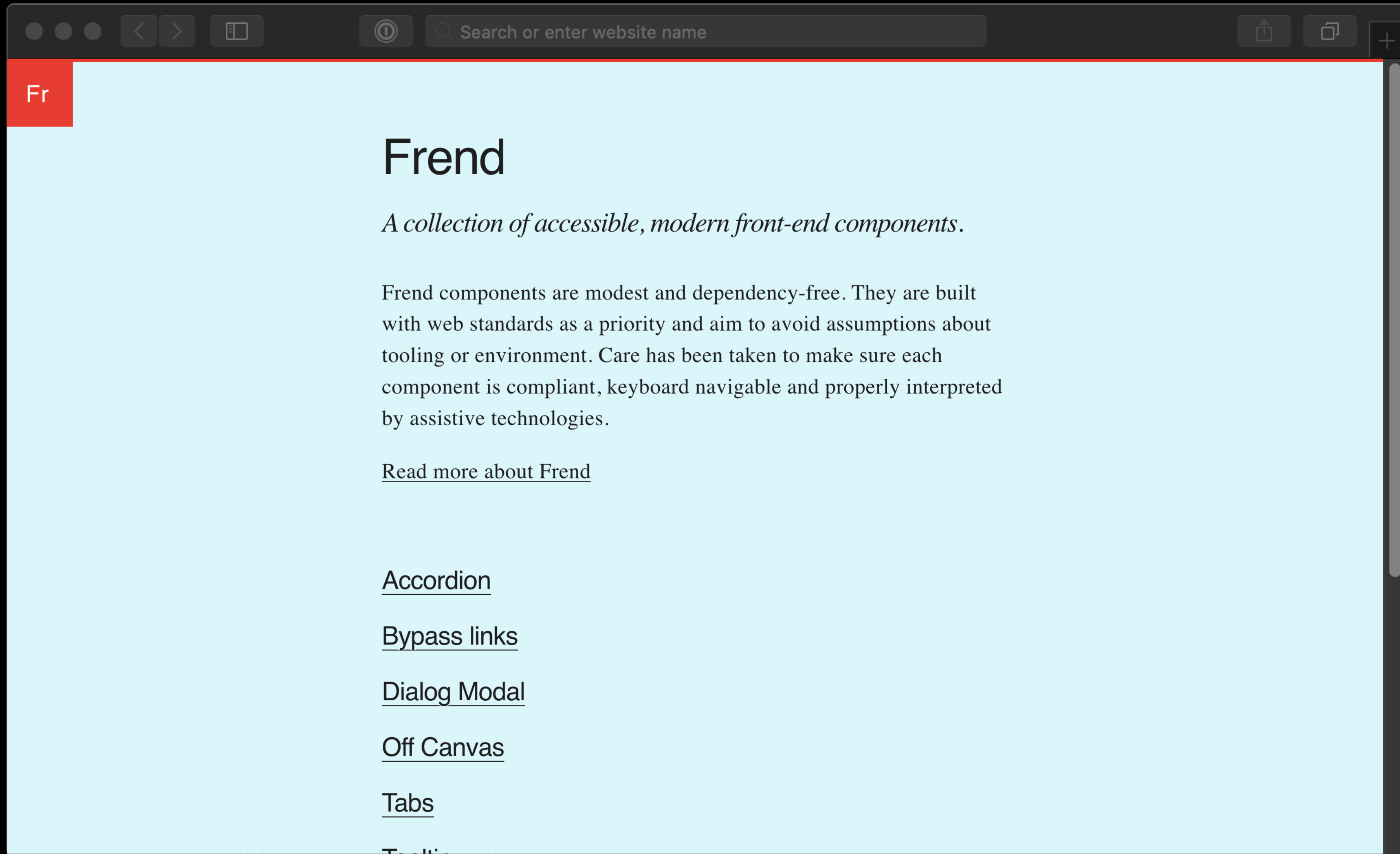
This page includes the following indexes of example implementations of [ARIA design patterns](#) included in [WAI-ARIA Authoring Practices 1.2](#).

- [Examples by Role](#)
- [Examples by Properties and States](#)

### Examples by Role

**NOTE:** The HC abbreviation means example has High Contrast support.

Role	Examples
<a href="#">alert</a>	<ul style="list-style-type: none"><li>• <a href="#">Alert</a></li><li>• <a href="#">Alert Dialog</a></li></ul>
<a href="#">alertdialog</a>	<a href="#">Alert Dialog</a>
<a href="#">article</a>	<a href="#">Feed</a>





<https://inclusive-components.design/>

# JavaScript & Accessibility

1. **Don't** use JavaScript for functionality HTML provides

2. Where possible, **don't** require JavaScript for critical features

1. **Do** use JavaScript to improve on the default accessible behaviour

2. **Do** use JavaScript to create components that don't exist



**“By default, HTML is accessible,** if used correctly.

Web accessibility involves ensuring that content remains accessible.”

“So next time someone tells you to make things accessible, tell them that instead **you don't intend to make it inaccessible in the first place.**”

— George Kemp

<https://dev.to/gkemp94/the-web-is-accessible-by-default-stop-breaking-it-4cg4>

# Thank you!

Ire Aderinokun 🇳🇮 🇬🇧

Independent **User Interface Engineer**

Google **Web Expert**

Co-Founder **Helicarrier**

[ireaderinokun.com](http://ireaderinokun.com)

[bitsofco.de](http://bitsofco.de)

[@ireaderinokun](https://twitter.com/ireaderinokun) / [@ire@front-end.social](https://twitter.com/ire@front-end.social) / [@ire.bsky.social](https://twitter.com/ire.bsky.social)

