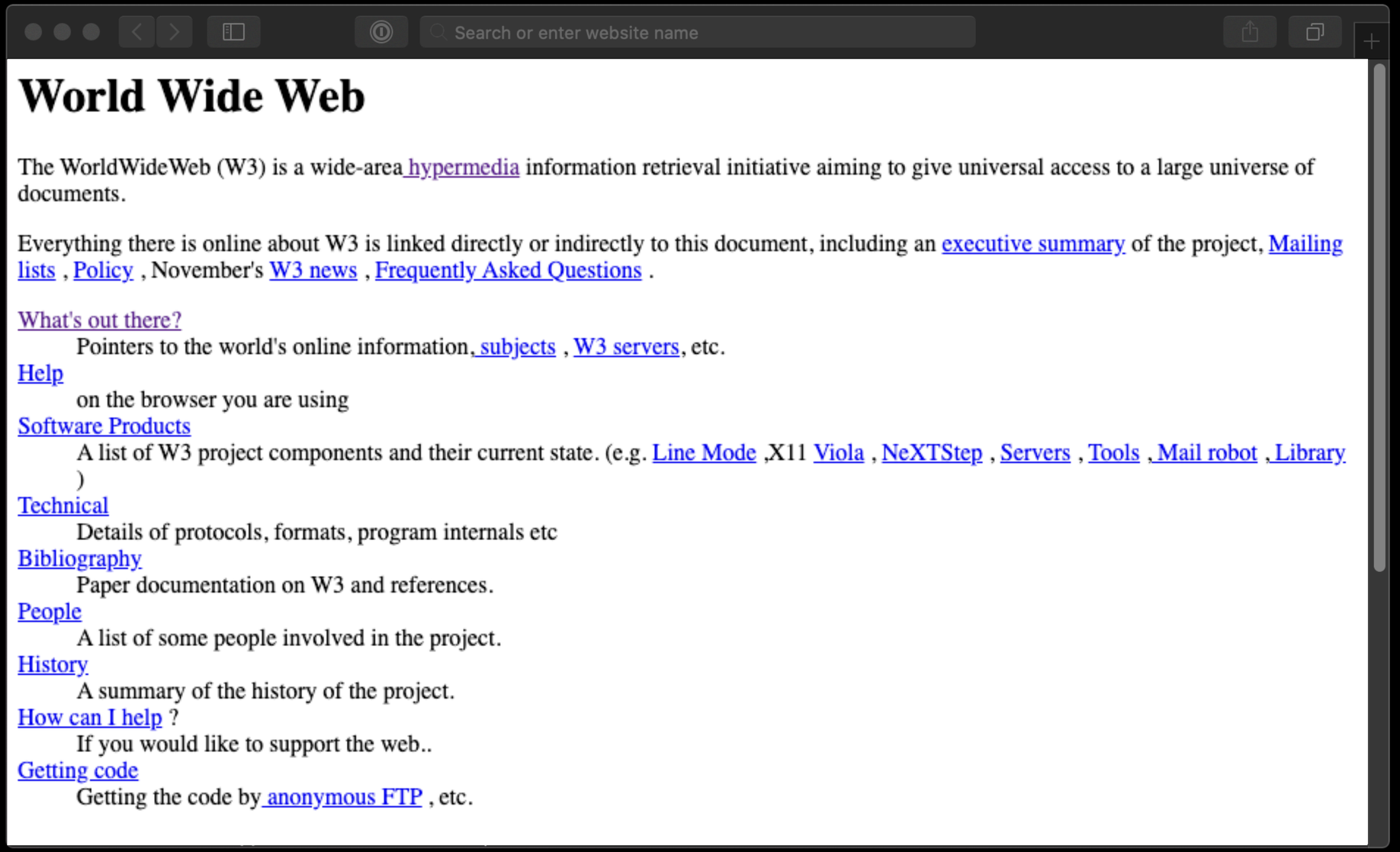
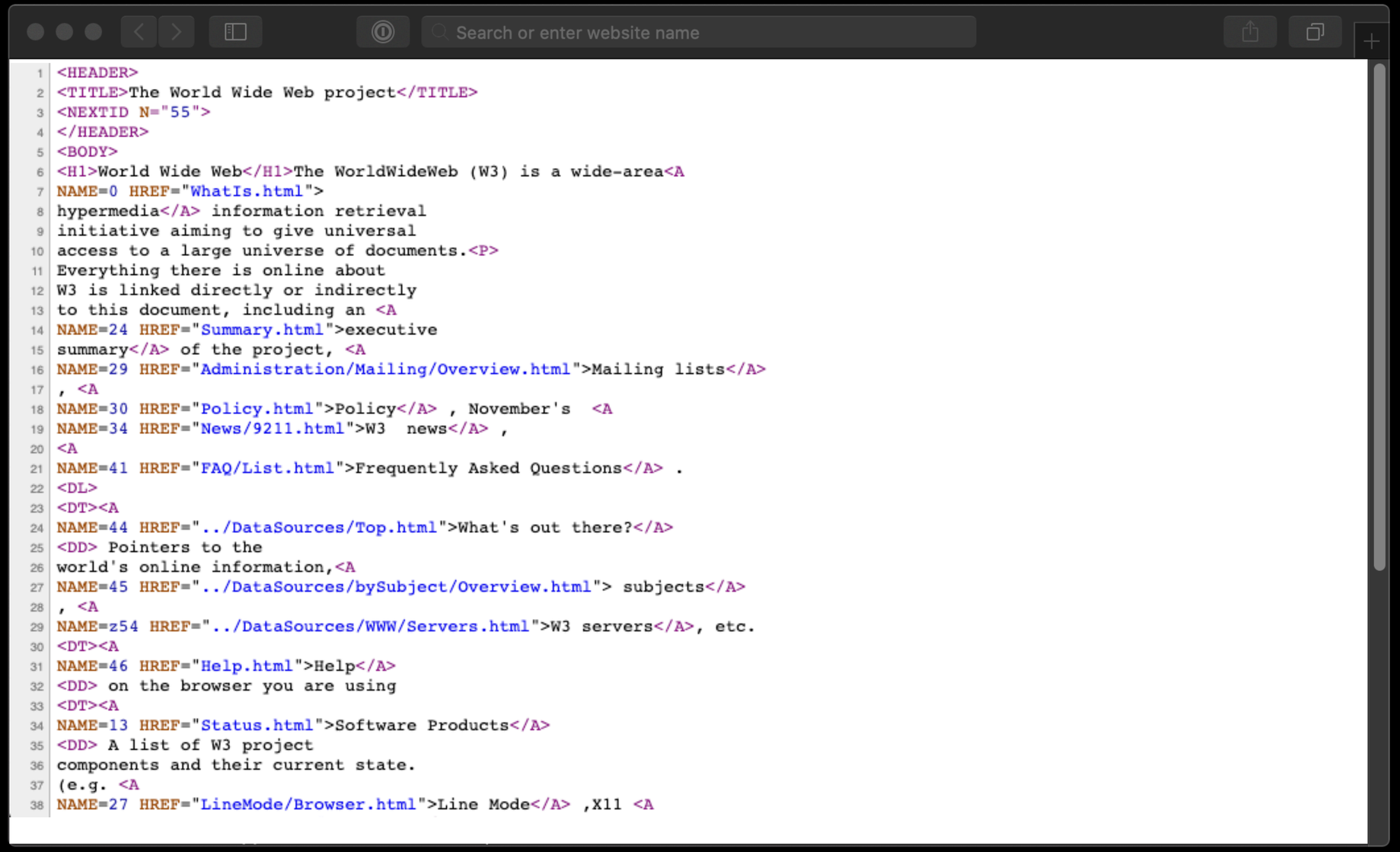
FUTURE-PROOF CSS

Ire Aderinokun @ An Event Apart “Front-End Focus” 2020



http://info.cern.ch/hypertext/WWW/TheProject.html



http://info.cern.ch/hypertext/WWW/TheProject.html

```
<HTML>

<HEAD></HEAD>

<HEADER>

    <TITLE>The World Wide Web project</TITLE>

    <NEXTID N="55">

</HEADER>

<BODY> ... </BODY>

</HTML>
```

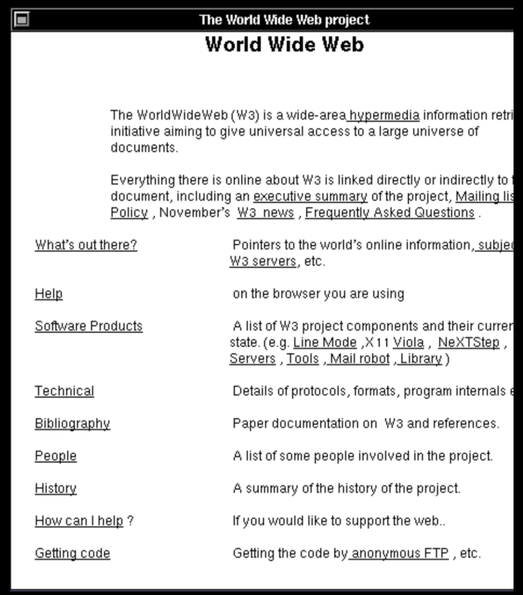
```
<HEADER>
    <TITLE>The World Wide Web project</TITLE>
    <NEXTID N=“55">
</HEADER>
<BODY> ... </BODY>
```

```
<HEADER>
    <TITLE>The World Wide Web project</TITLE>
    <NEXTID N=“55">
</HEADER>
<BODY> ... </BODY>
```

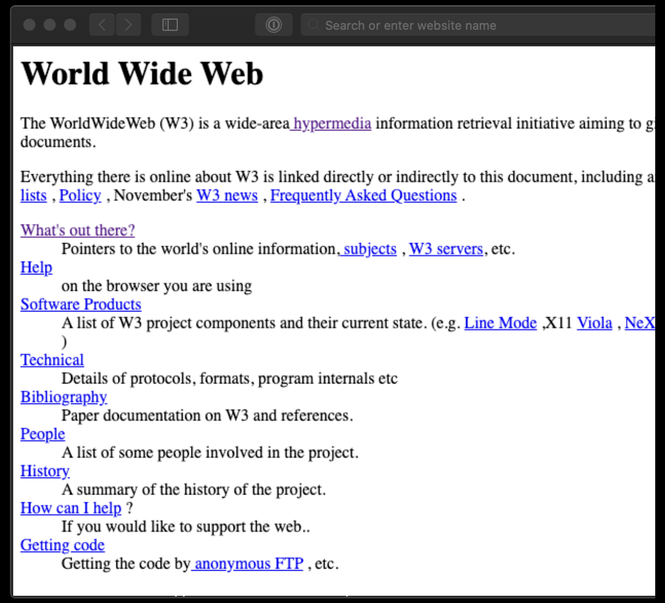
<nextid> is an obsolete HTML element that served to enable the NeXT web designing tool to generate automatic NAME labels for its anchors. It was generated by that web editing tool automatically and was not to be adjusted or entered by hand. It is also probably one of the least understood of all of the early HTML elements.

— MDN Web Docs

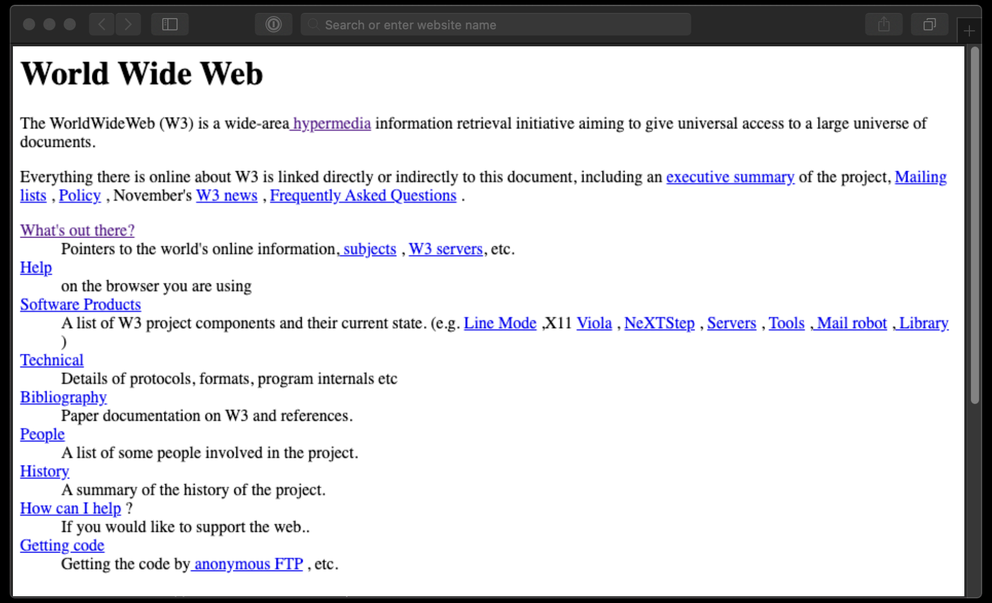
https://developer.mozilla.org/en-US/docs/Web/HTML/Element/nextid



1991



1998



2020

DON'T BREAK THE WEB

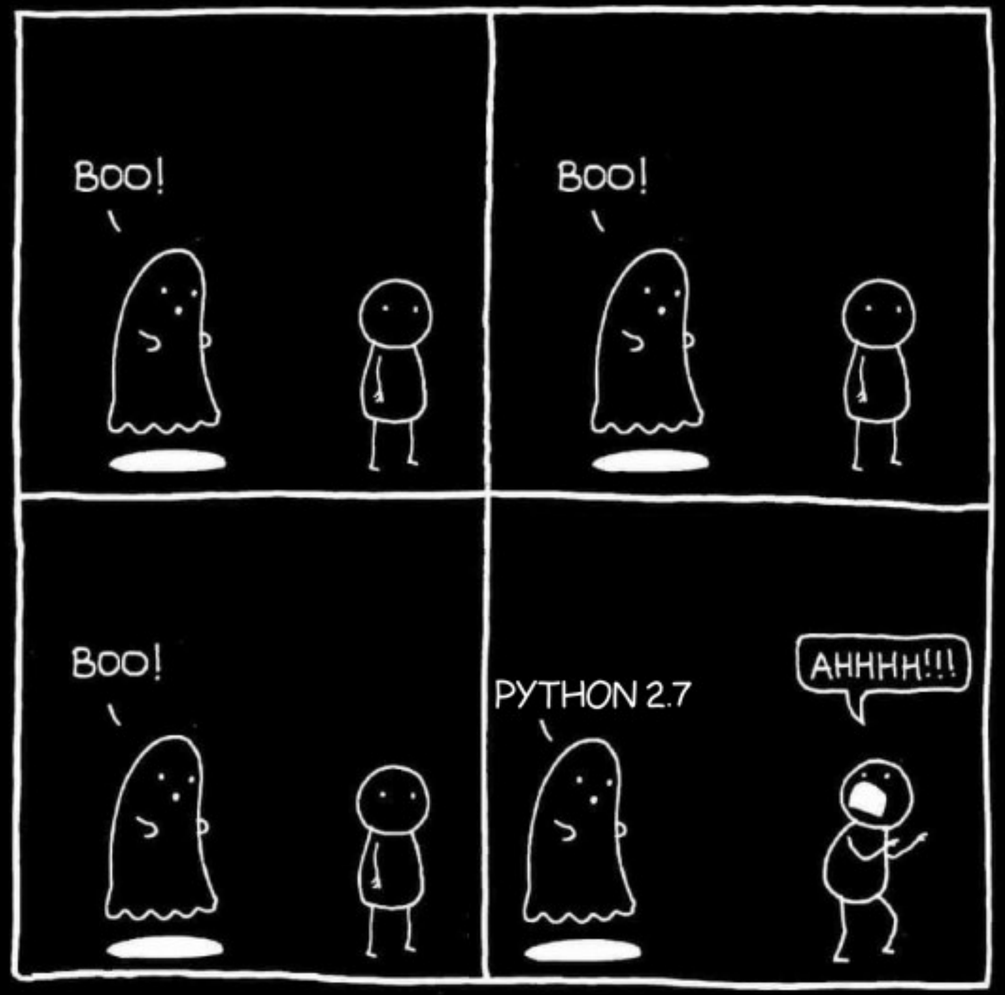
Web technologies won’t change in a way that breaks previously compliant websites

DIAMONDS

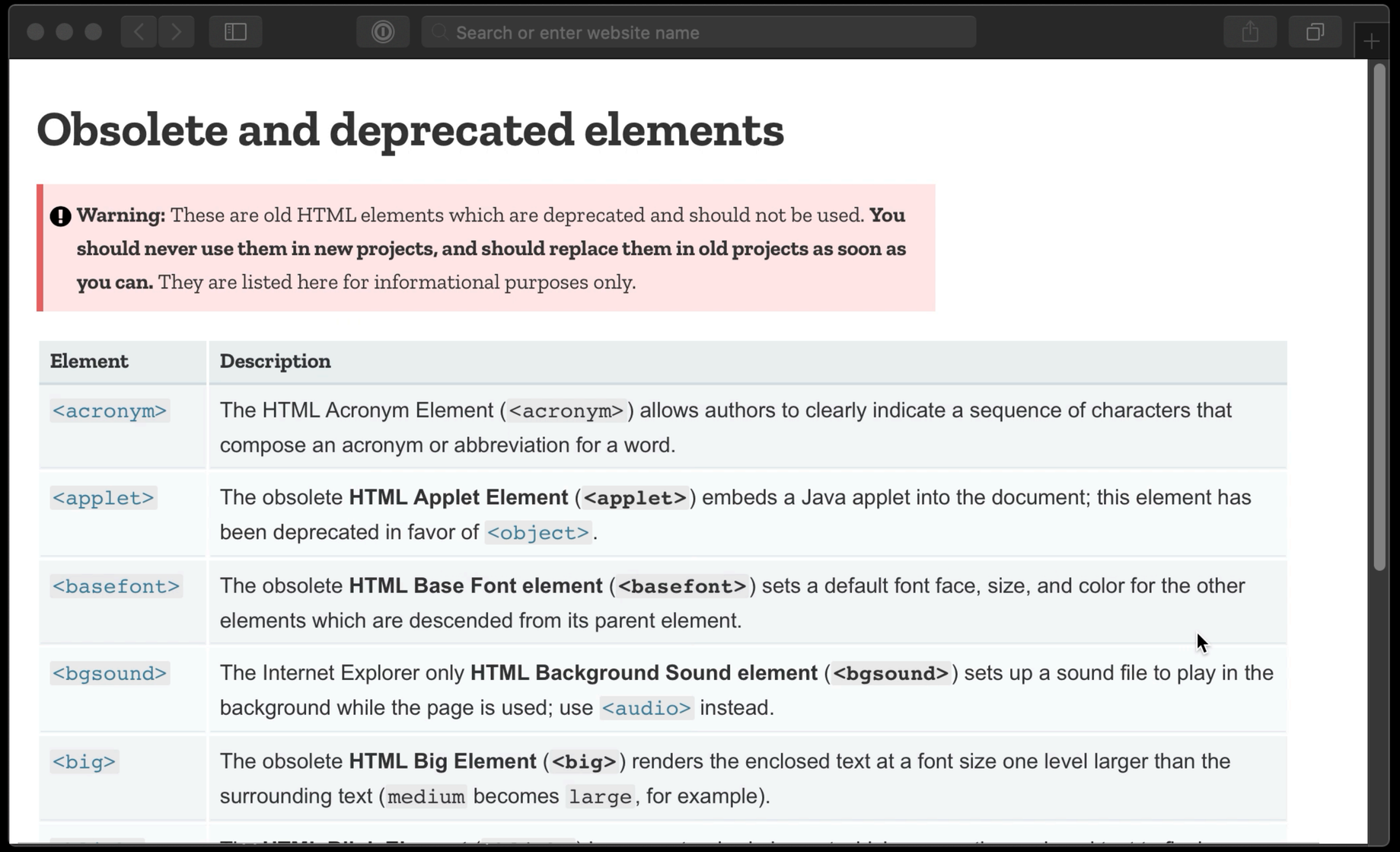
"I LAST FOREVER"

WEBSITES

"HOLD MY BEER"


BOO!
BOO!
BOO!
PYTHON 2.7
AHHHH!!!


Boo!
Boo!
Boo!
CSS 500!



https://developer.mozilla.org/en-US/docs/Web/HTML/Element

But websites still break 💔

Web technologies won’t change in a way that breaks previously compliant websites

FUTURE-PROOF CSS !

WHAT WILL CSS LOOK LIKE IN THE FUTURE ?

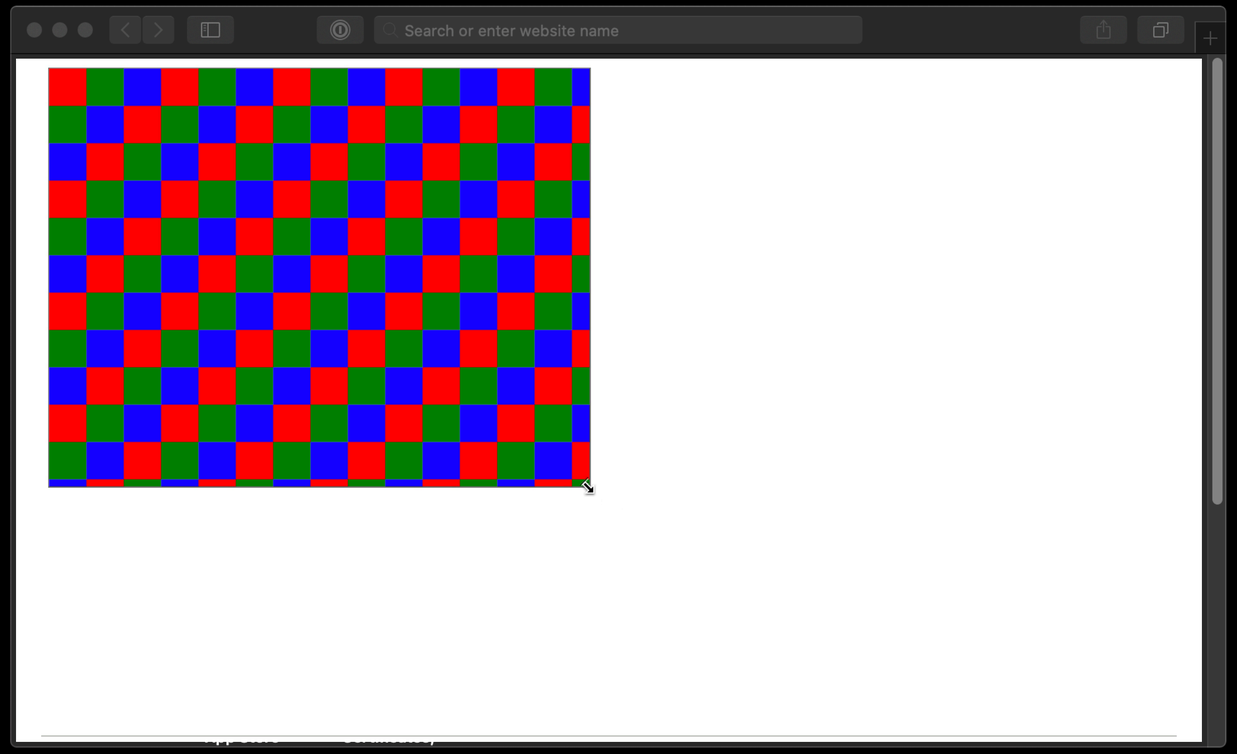

MORE FEATURES!

JS in CSS? 🤡

```
main {

    display: 'serviceWorker' in navigator ? "…" : "…";

}
```

Houdini

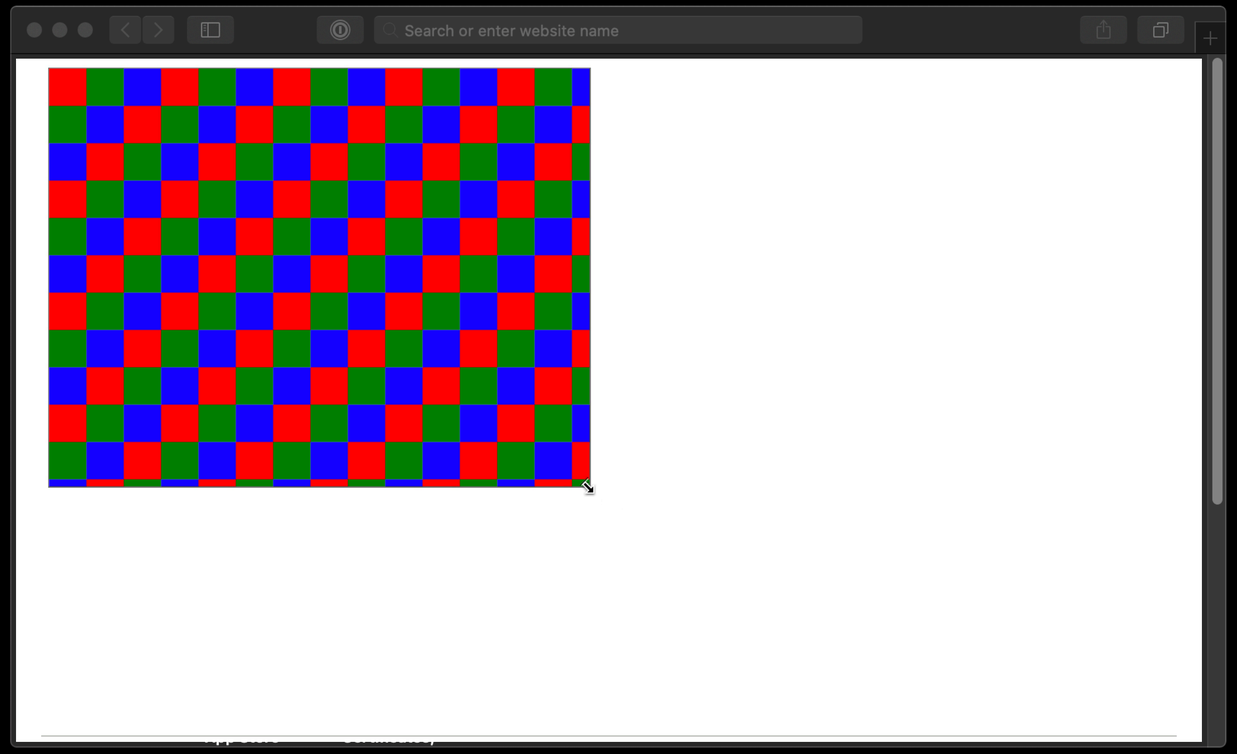
```
textarea {

    background-image: paint(checkerboard);

}
```



https://googlechromelabs.github.io/houdini-samples

Houdini

```
class CheckerboardPainter {

    paint(ctx, geom, properties) {

        /* implement checkerboard */

    }

}

registerPaint('checkerboard', CheckerboardPainter);
```



https://googlechromelabs.github.io/houdini-samples

JS in CSS

Write Javascript in your CSS!

Chrome for TV	Edge	Firefox	Chrome	WatchOS Safari	iOS Safari	Opera MAXI	Chrome for Watch	Android Browser	Samsung Internet
8	79	75	79	12	13.1			4.3	9.2
9	80	76	80	12.1	13.2			4.4	10.1
10	81	77	81	13	13.3			4.4.4	11.2
11	83	78	83	13.1	13.5	all	81	81	12.0
		79	84	14	14.0				
		80	85	TP					


PROGRESSIVE ENHANCEMENT
The Book of
PSALMS

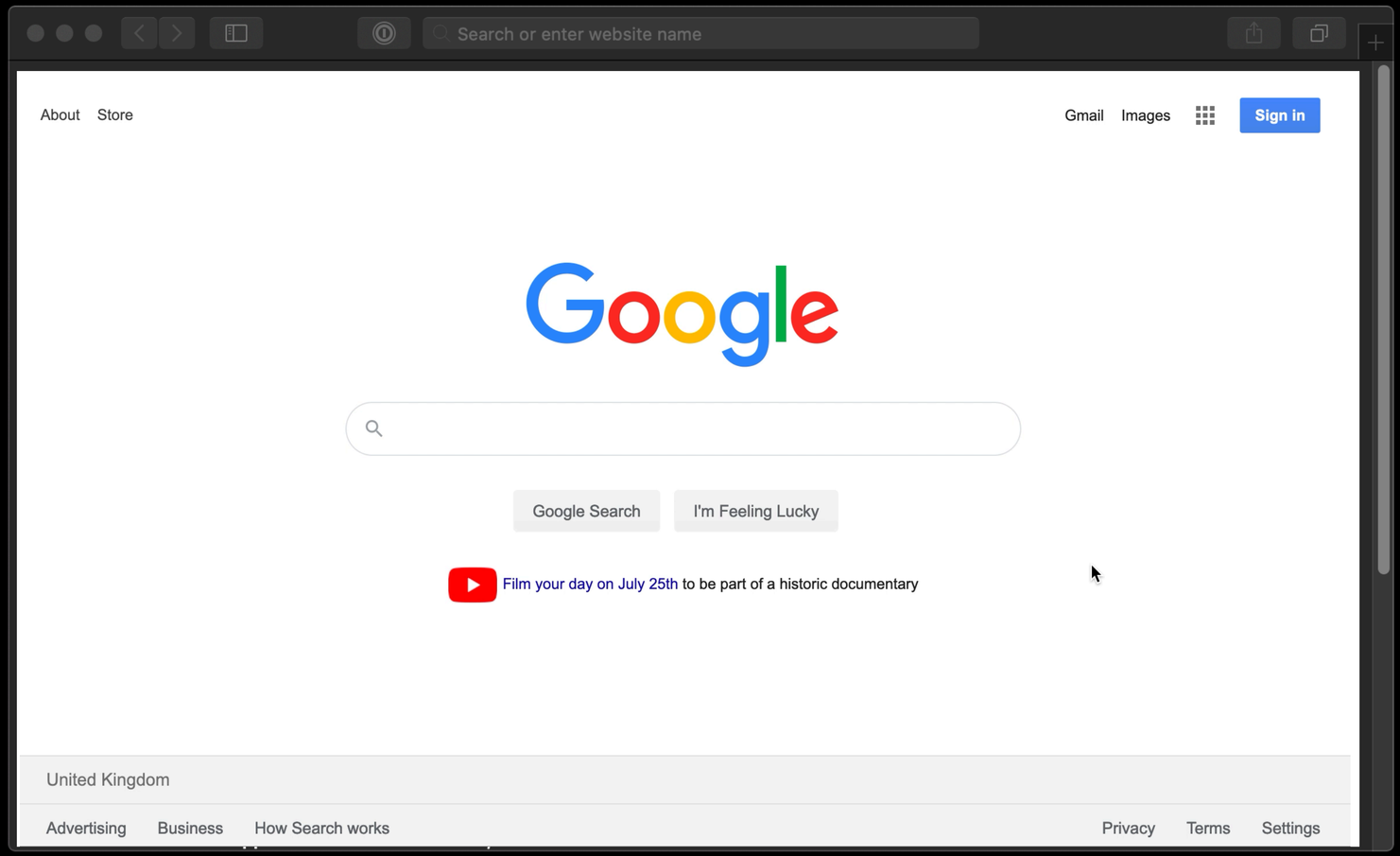
Progressive enhancement builds documents for the least capable or differently capable devices first, then moves on to enhance those documents with separate logic for presentation, in ways that don’t place an undue burden on baseline devices but which allow a richer experience for those users with modern graphical browser software

— Stephen Champeon (2003)

http://hesketh.com/publications/inclusive_web_design_for_the_future/

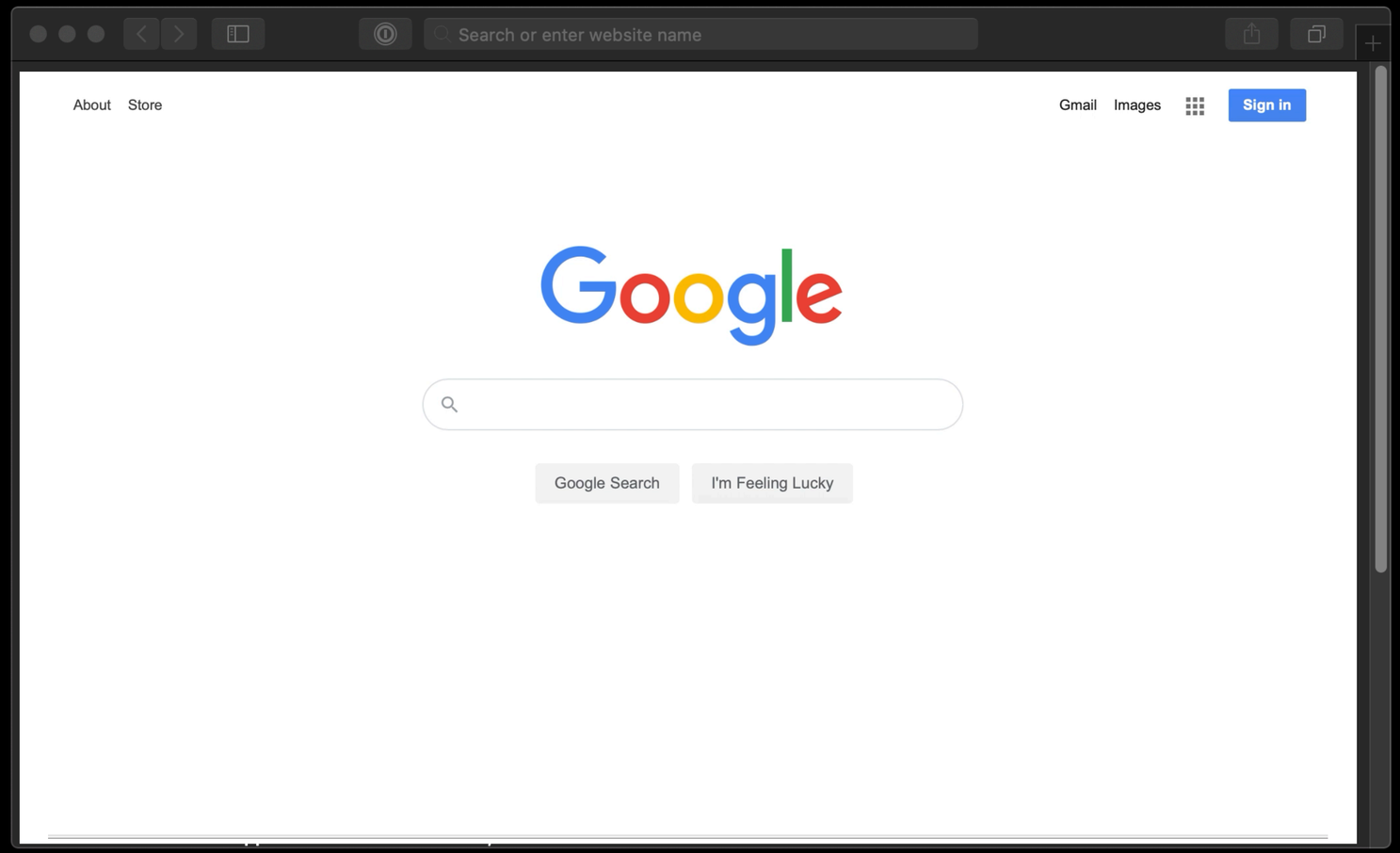
Graceful degradation is the practice of building your web functionality so that it provides a certain level of user experience in more modern browsers, but it will also degrade gracefully to a lower level of user experience in older browsers.

Full experience



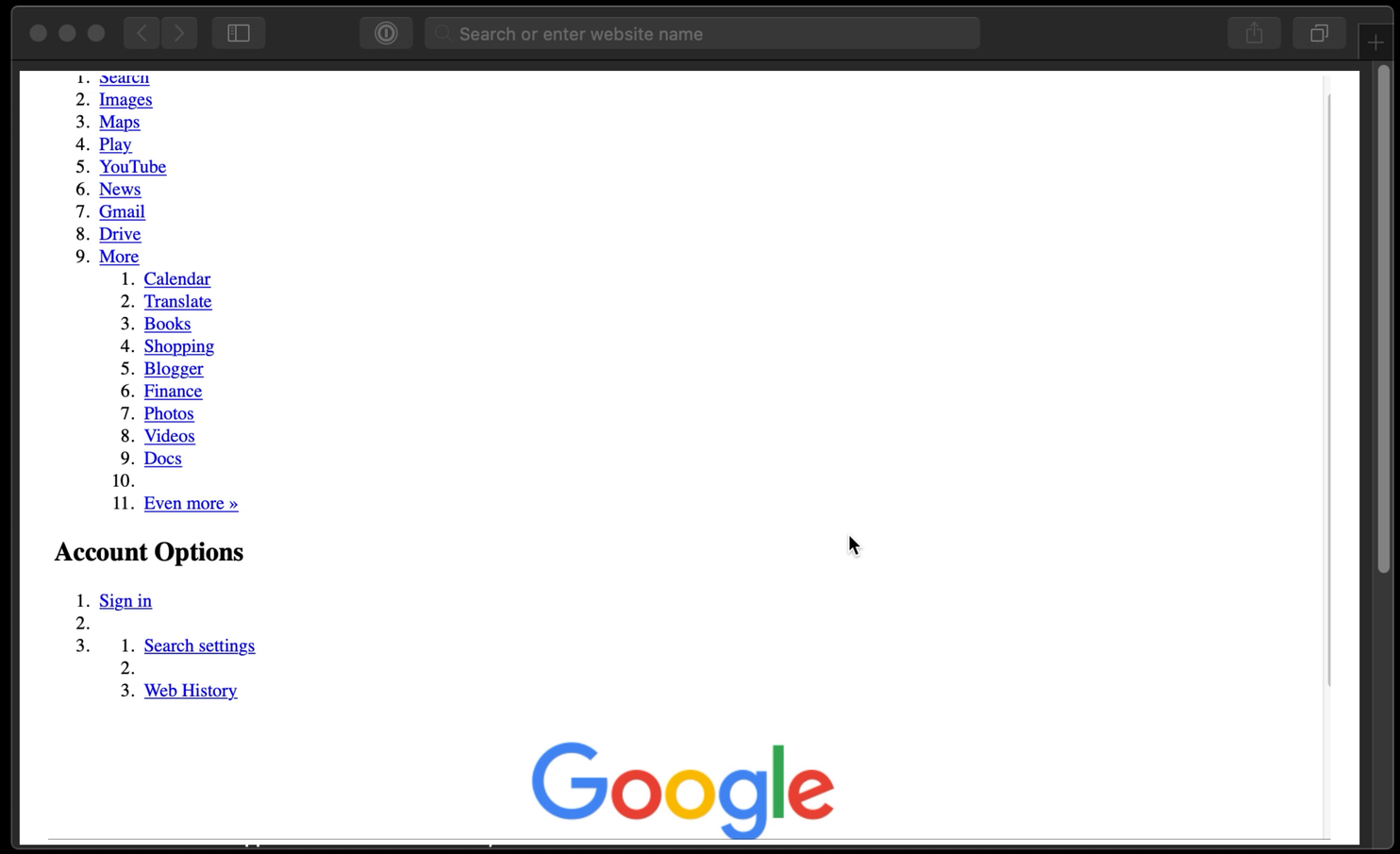
https://www.google.com

Experience with Javascript disabled



https://www.google.com

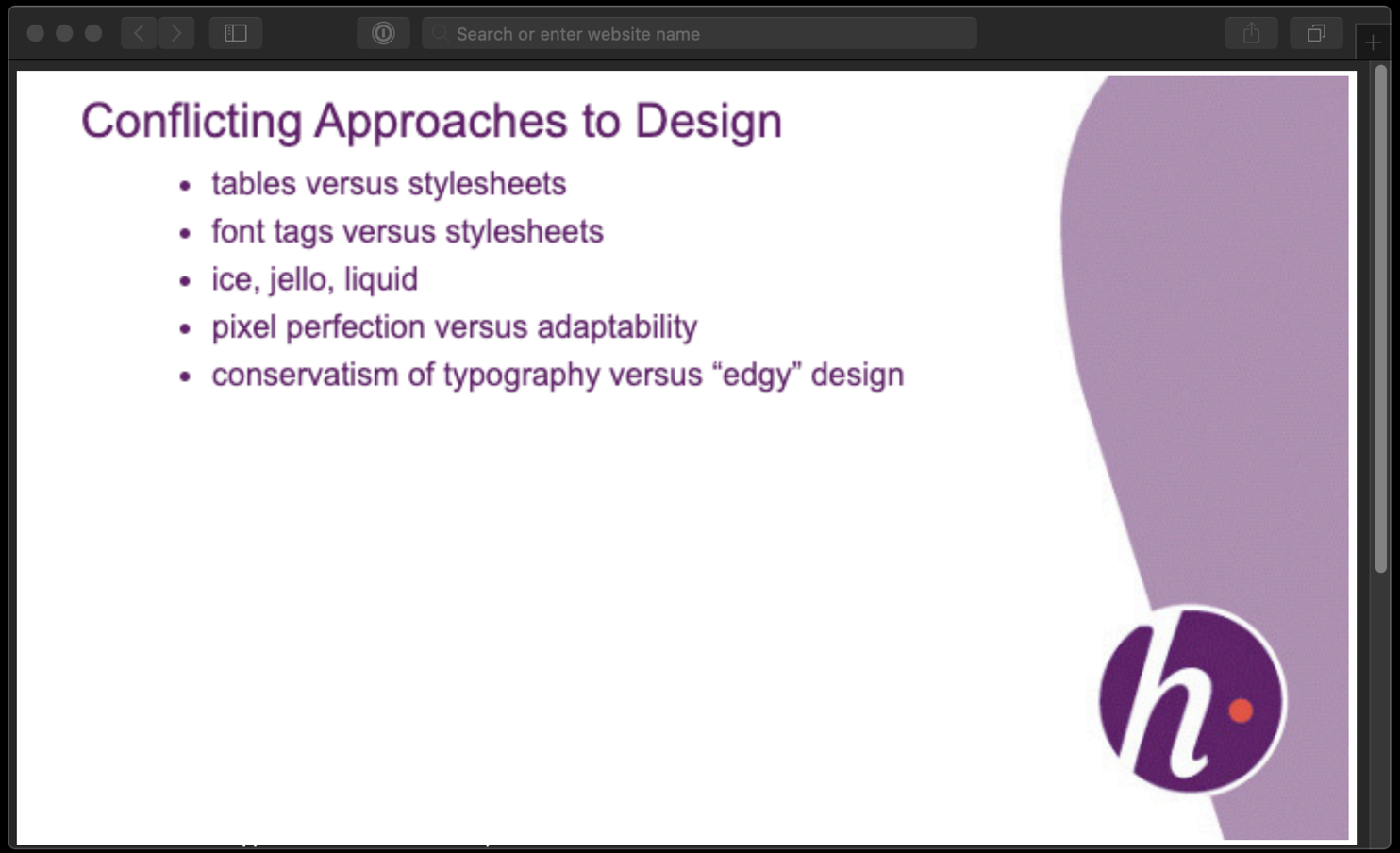
Experience with both Javascript and Stylesheets disabled



https://www.google.com

Example of a graceful degradation approach

The problems are different, but the solution is the same



http://hesketh.com/publications/inclusive_web_design_for_the_future

1. Start by providing a core experience to all browsers

2. Add styling and functionality to enhance the experience for more capable browsers

But, progressive enhancement with CSS is kinda hard

JS already has the right tools

✅ Feature detection

✅ Error handling

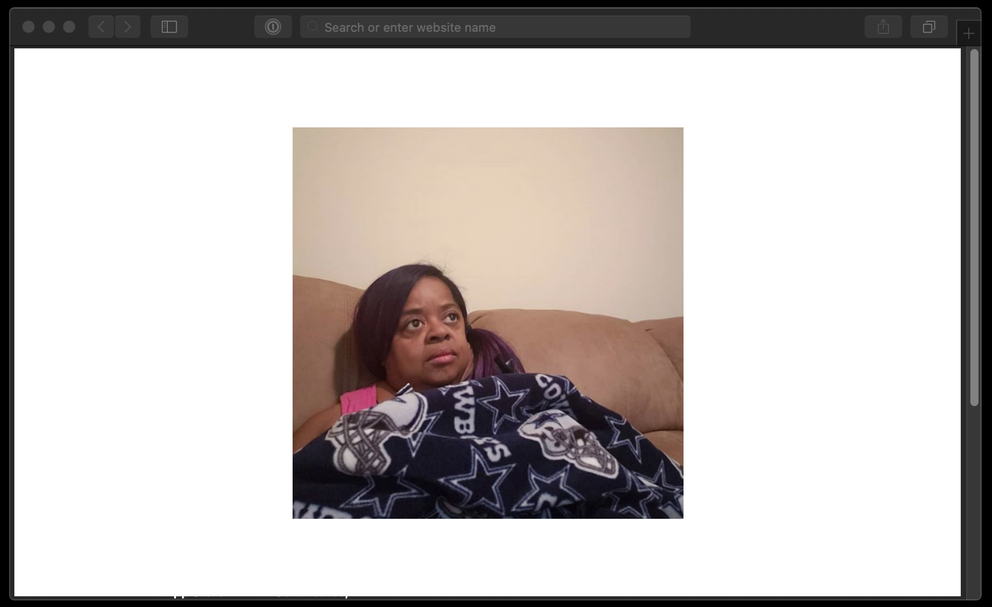
```
if (‘serviceWorker’ in navigator) {
    navigator.serviceWorker.register()
        .then(() ⇒ { ... })
        .catch(() ⇒ { ... })
} else {
    // do something else
}
```

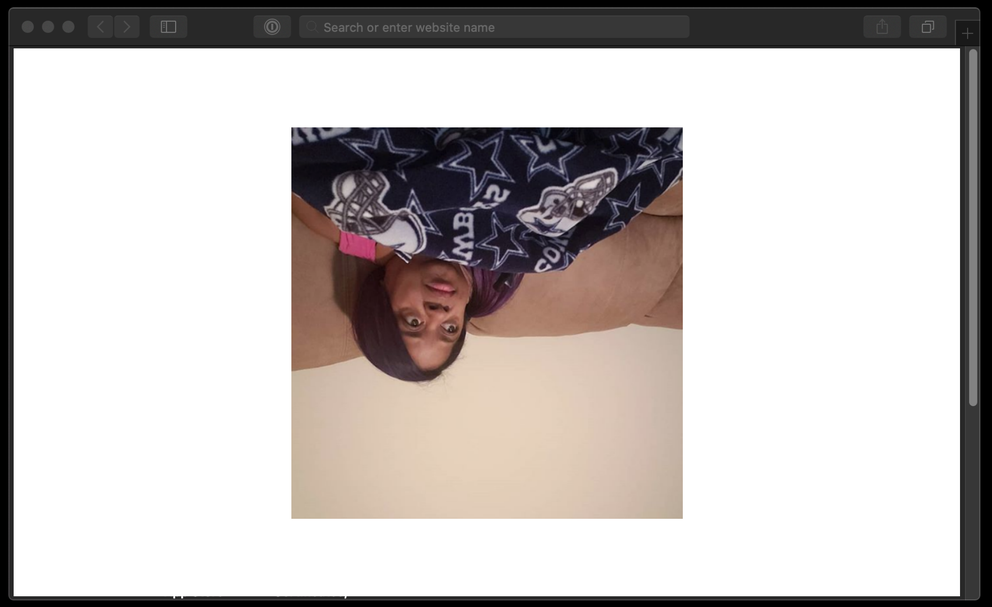
```
if (‘serviceWorker’ in navigator) {
    navigator.serviceWorker.register()
        .then(() ⇒ { ... })
        .catch(() ⇒ { ... })
} else {
    // do something else
}
```

CSS is lacking the right tools

❌ Feature detection (sort of)

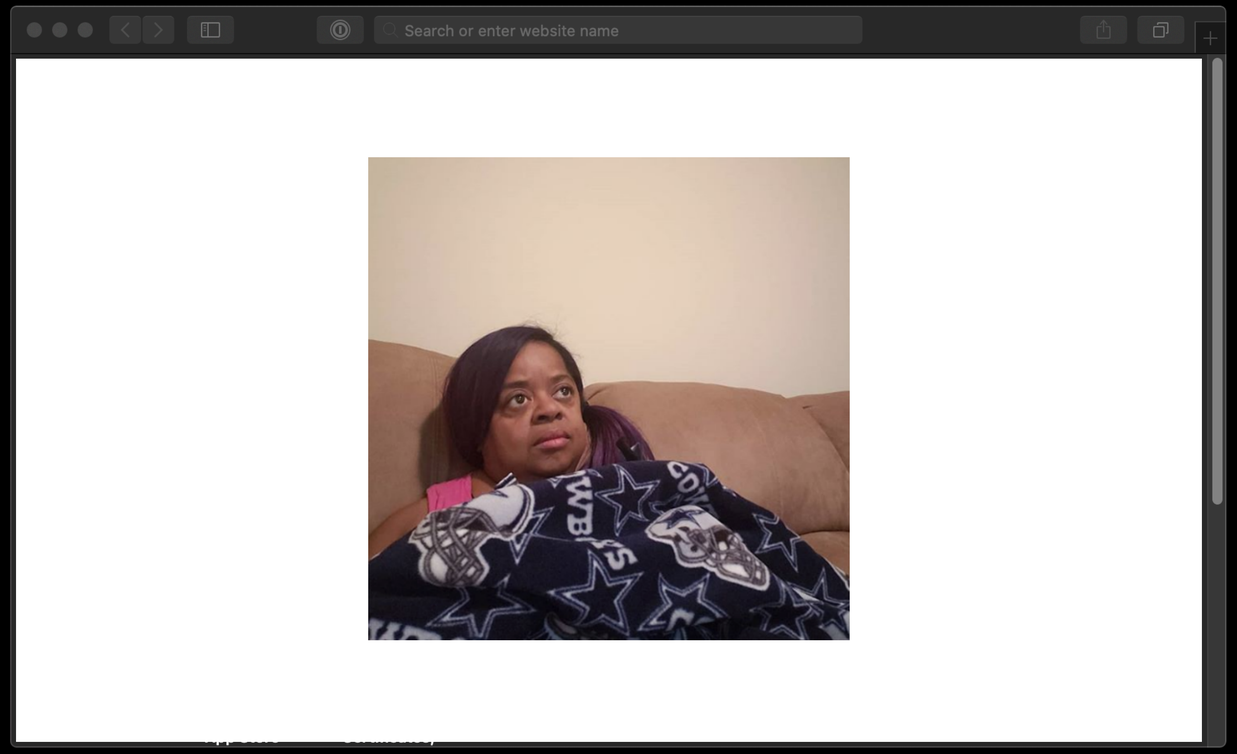
❌ Error handling


Search or enter website name


Search or enter website name

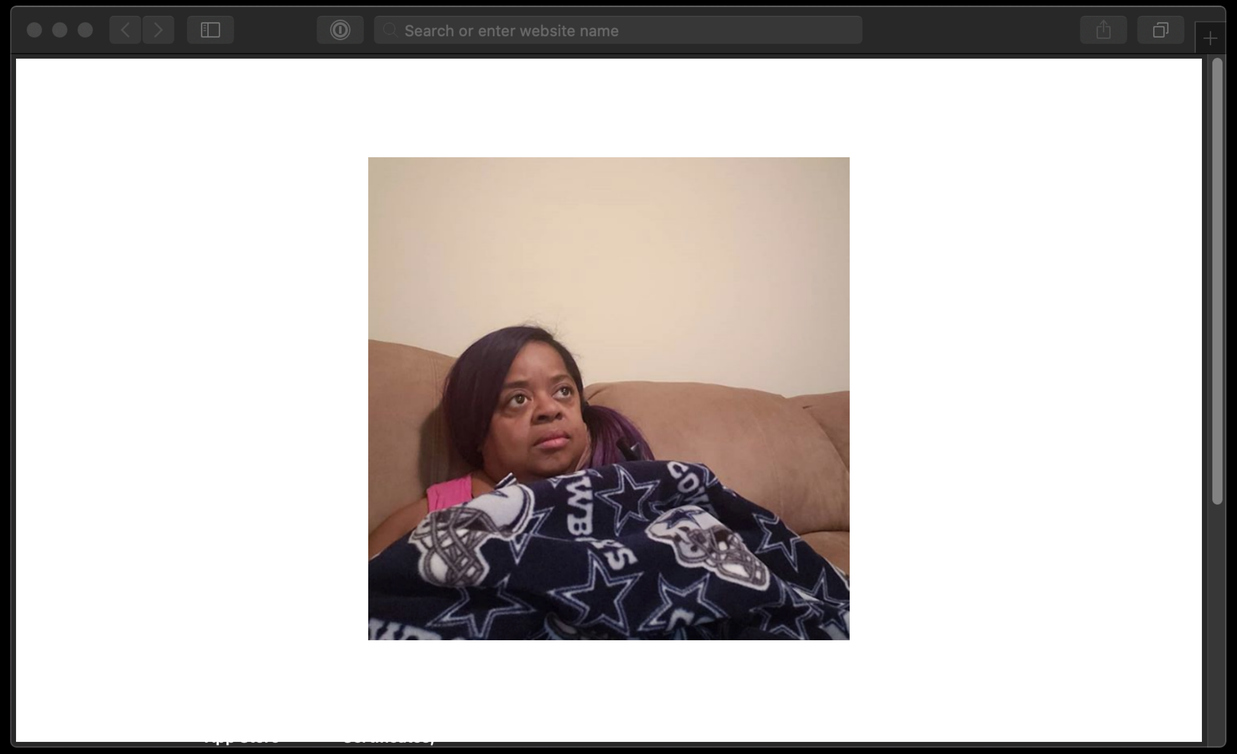
Faulty selector ☹️

```
image {

    transform: rotate(180deg);

}
```



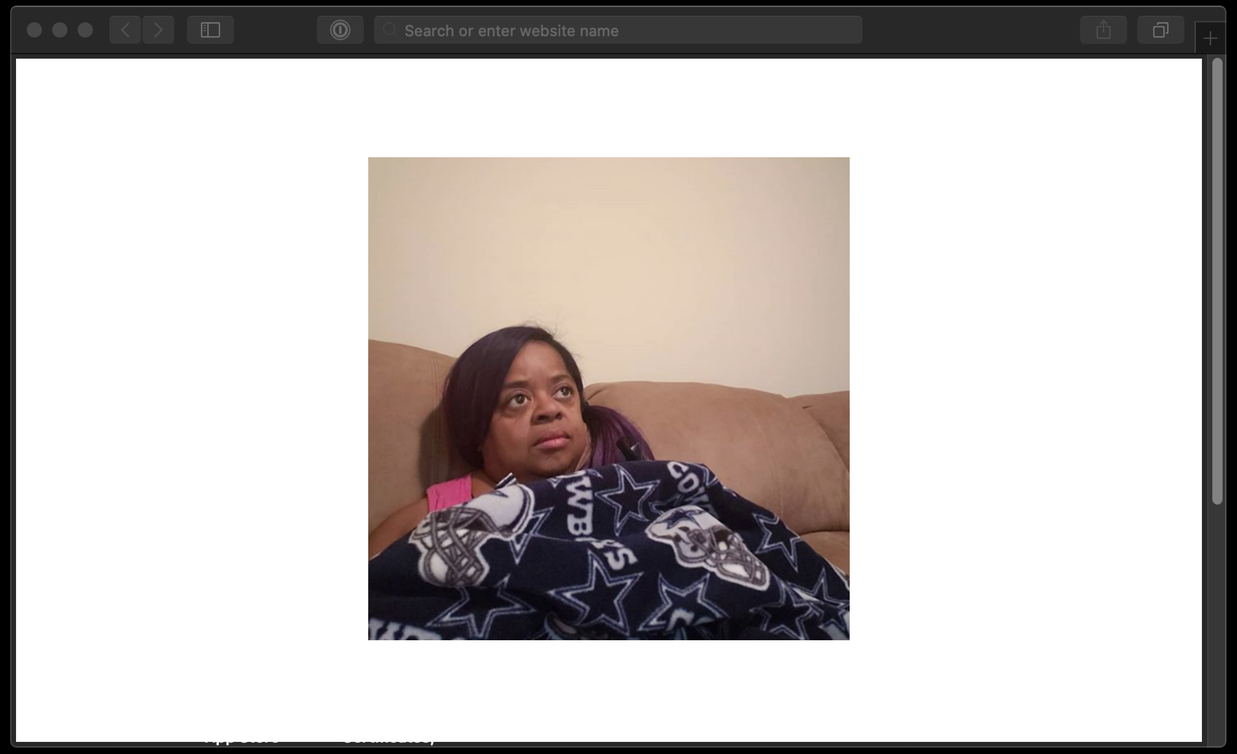
Faulty property ☹️

```
img {

    transformer: rotate(180deg);

}
```



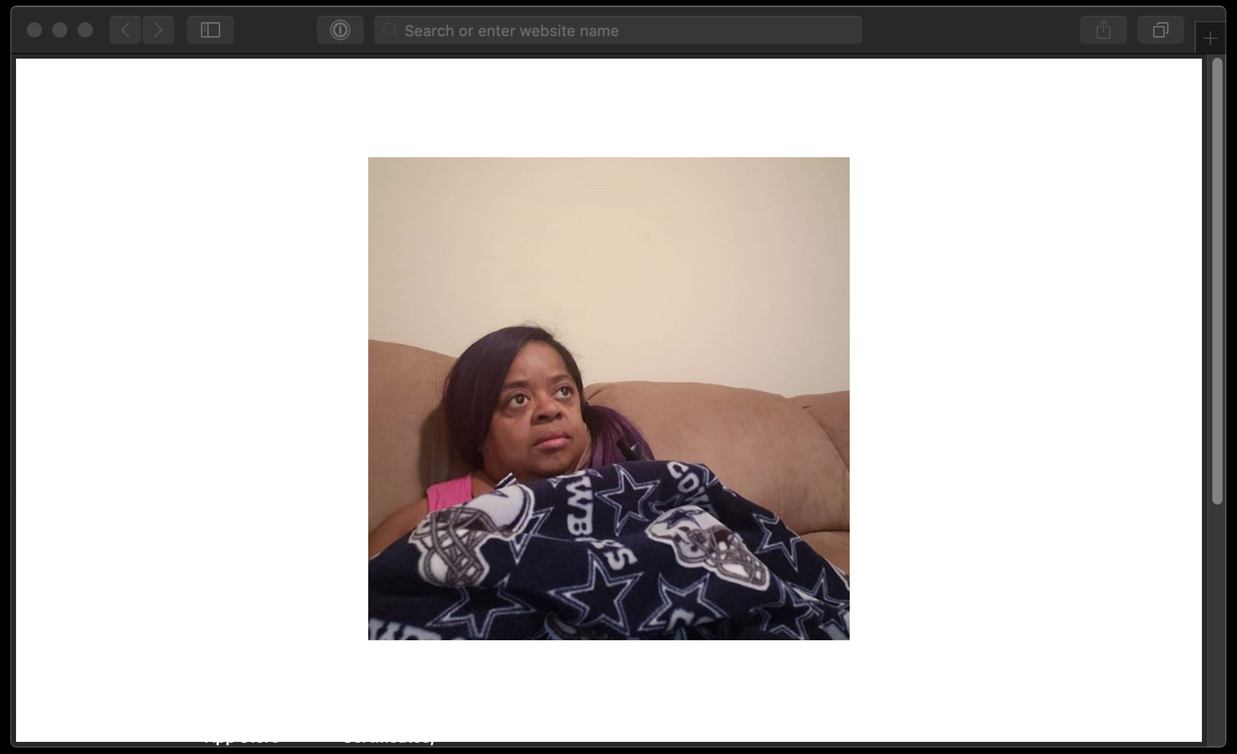
Faulty value ☹️

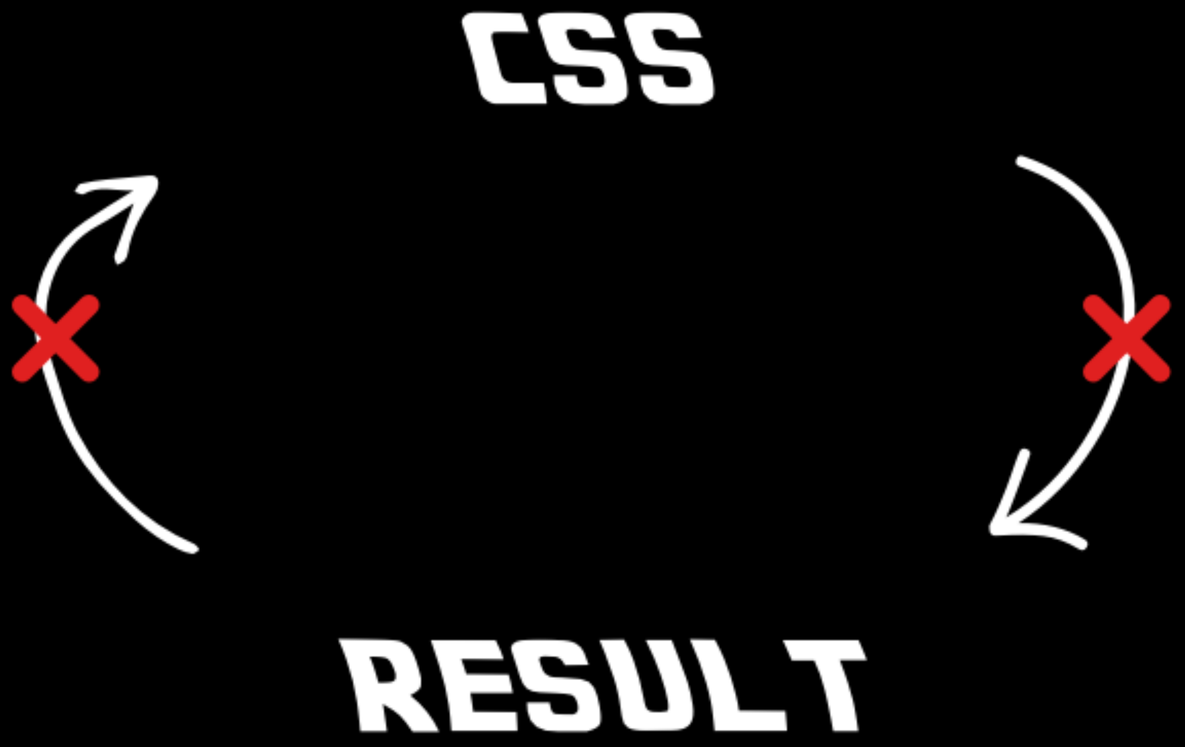
```
img {

    transform: flip(180deg);

}
```



Unsupported browser ☹️

```
img {

    transform: rotate(180deg);

}
```




CSS
RESULT

Progressive enhancement with CSS is kinda hard, but still doable

Don’t use CSS (if you don’t have to)

Start with sensible HTML

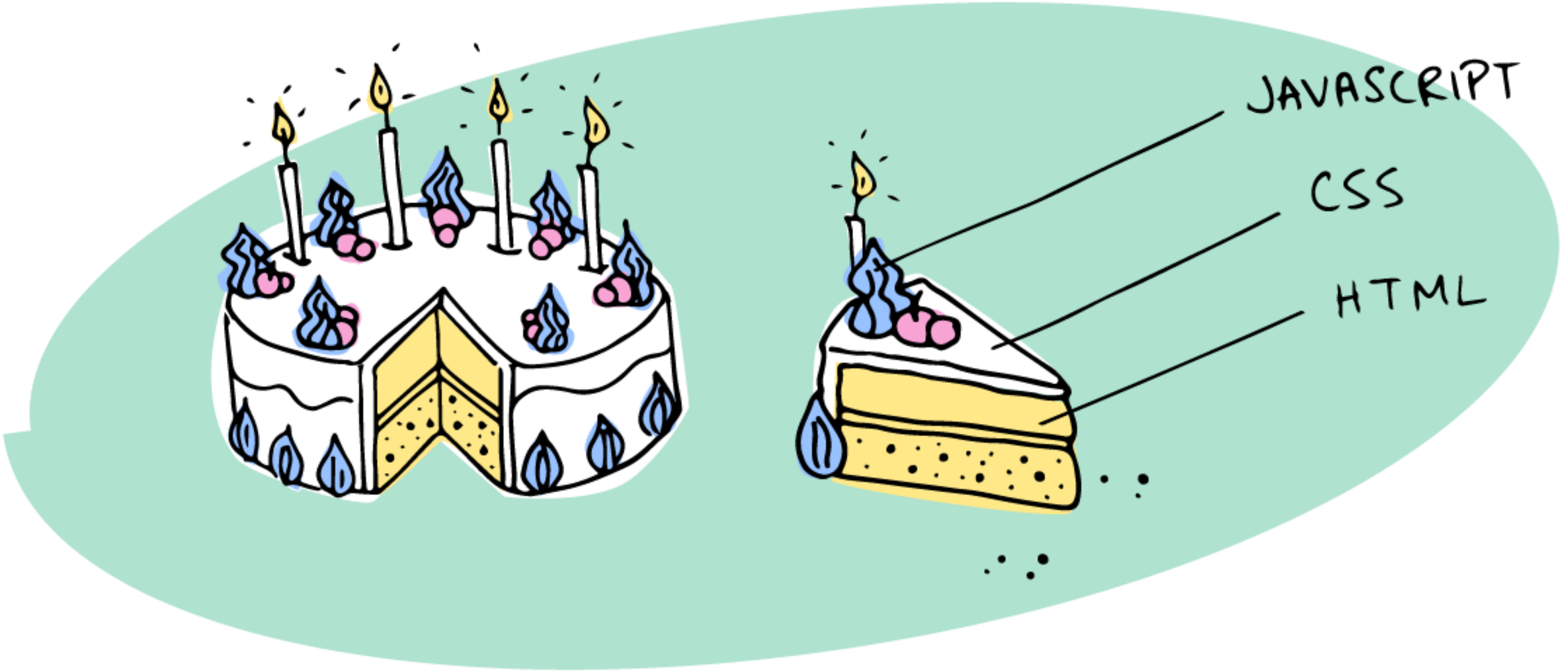


Image from https://www.shopify.com/partners/blog/what-is-progressive-enhancement-and-why-should-you-care

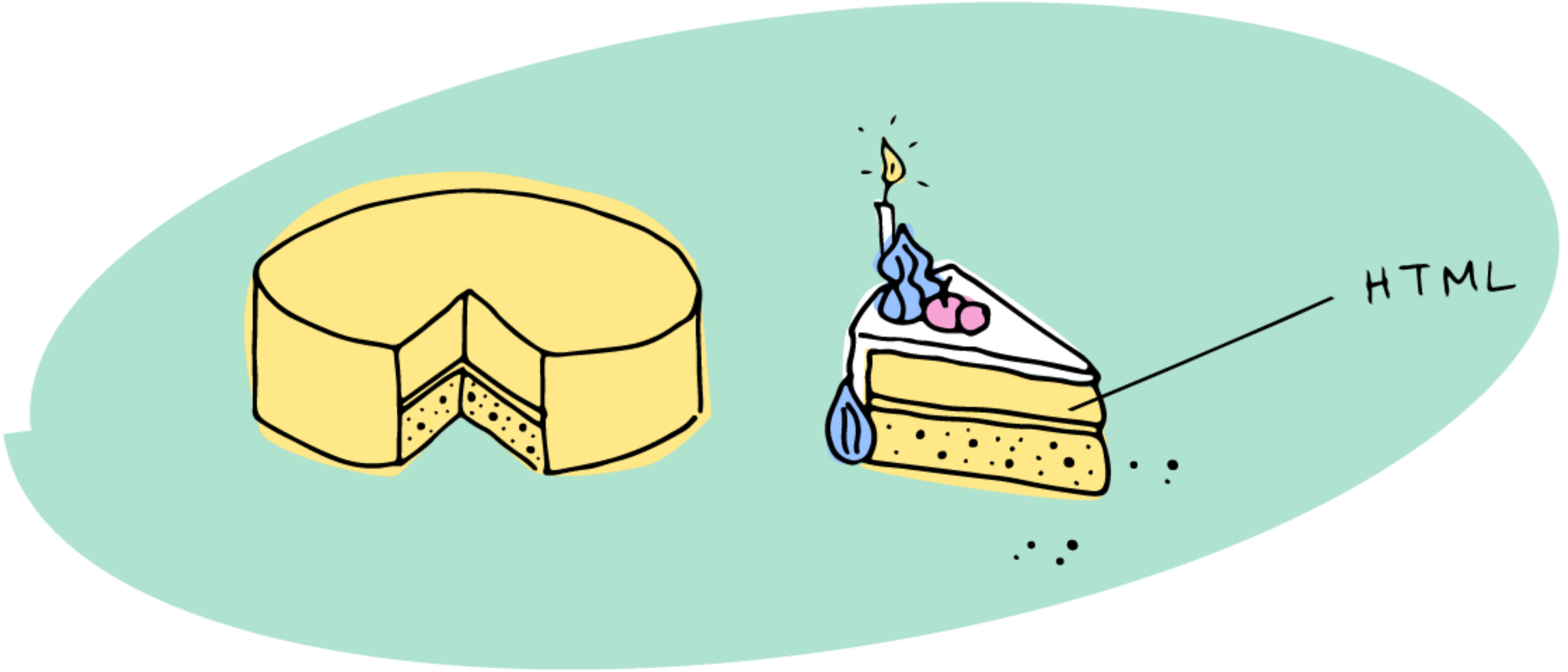
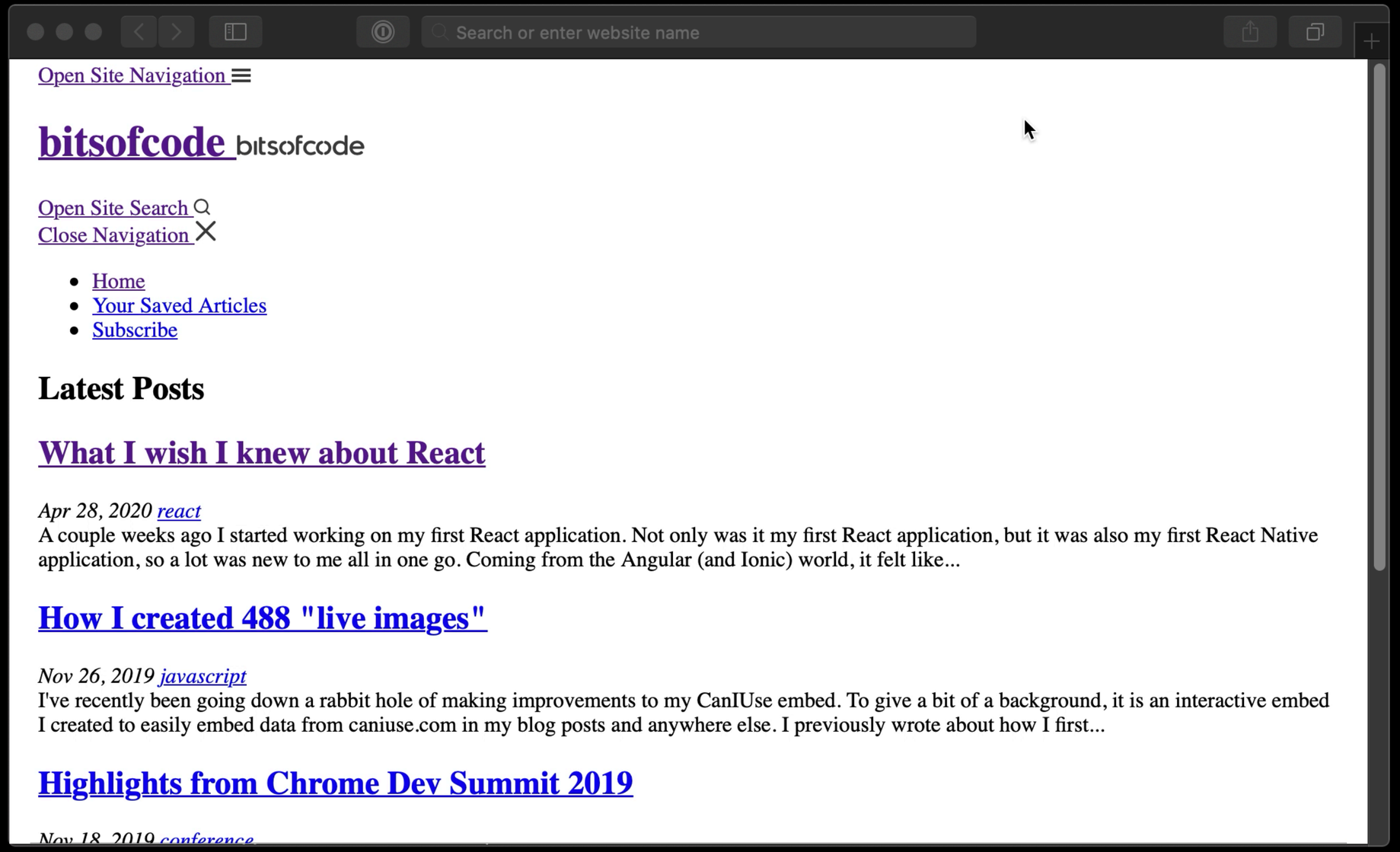


Image from https://www.shopify.com/partners/blog/what-is-progressive-enhancement-and-why-should-you-care

Make sure the document works
without CSS in the first place


Search or enter website name
Open Site Navigation
bitsofcode bitsofcode
Open Site Search
Close Navigation
Home
Your Saved Articles
Subscribe
Latest Posts
What I wish I knew about React
Apr 28, 2020 react
A couple weeks ago I started working on my first React application. Not only was it my first React application, but it was also my first React Native application, so a lot was new to me all in one go. Coming from the Angular (and Ionic) world, it felt like...
How I created 488 "live images"
Nov 26, 2019 javascript
I've recently been going down a rabbit hole of making improvements to my CanIUse embed. To give a bit of a background, it is an interactive embed I created to easily embed data from caniuse.com in my blog posts and anywhere else. I previously wrote about how I first...
Highlights from Chrome Dev Summit 2019
Nov 18, 2019 conference

```
<body>

    <footer>...</footer>

    <nav>...</nav>

    <header>...</header>

    <main>...</main>

</body>
```

```
body { display: flex }

header { order: 1 }

nav { order: 2 }

main { order: 3 }

footer { order: 4 }
```

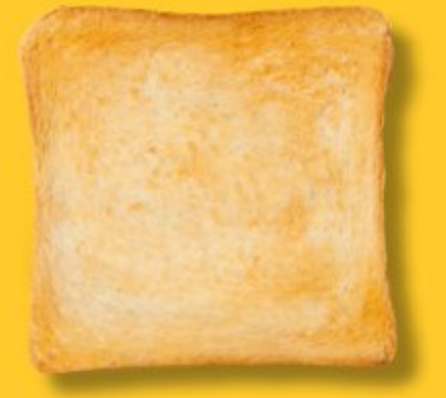
✅

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

```
body { display: flex }
```

Use the cascade

DIVIDING WEB DEVELOPERS SINCE 1994



The cascade is built-in progressive enhancement

```
.element {
    -webkit-transition: all 4s ease;
    -moz-transition: all 4s ease;
    -ms-transition: all 4s ease;
    -o-transition: all 4s ease;
    transition: all 4s ease;
}
```

```
body {
  background: url(fallback.png);
  background-image: url(awesome.svg), none;
}
```

Write mobile-first CSS

min-width > max-width

On a mobile device 📱

```
main { width: 800px }
```

```
@media (max-width: 900px) {
    main { width: 100% }
}
```

```
main { width: 100% }
```

```
@media (min-width: 900px) {
    main { width: 800px }
}
```


On average, mobile devices are slower
and less capable than desktop devices

Mobile styles are more adaptable

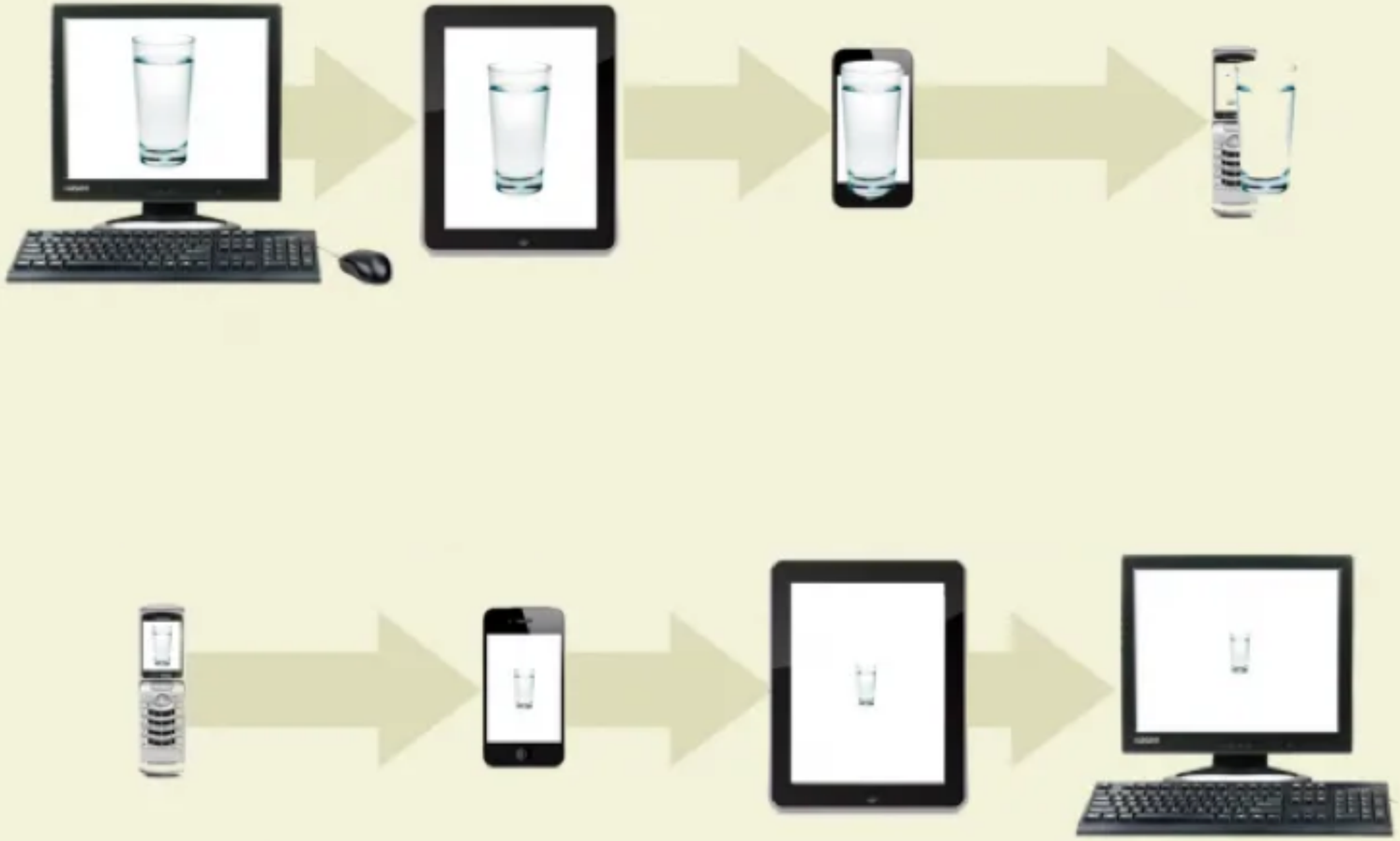


Image from https://bradfrost.com/blog/post/mobile-first-responsive-web-design

Will only work on larger screen sizes

```
main {

    width: 800px;

}
```

Will work on any device and screen size

```
main {

    width: 100%;

}
```

Don’t be afraid to use the “old thing”

doineedtouse > caniuse

Item title

Item description

Grid

2-Column

A (one half)	B (one half)

3-Column

A (one third)	B (one third)	C (one third)
D (two thirds)		E (one third)

4-Column

A (one fourth)	B (one fourth)	C (one fourth)	D (one fourth)
E (three fourths)			F (one fourth)

Reversed

Float-based layout 😮

```
.column {

  position: relative;

  width: 100%

  min-height: 1px;

  padding: 0px $small-space;

}
```

```
.column

[dir="rtl"] .reverse .column {

  float: left;

}
```

It just works

The goal of web design is not merely to dazzle, but to deliver information to the widest audience

— Stephen Champeon

http://hesketh.com/publications/inclusive_web_design_for_the_future/

Ok, but what if I do need to use the "new thing”? 🥺

Use feature queries!

CSS is lacking the right tools

✅ Feature detection (sort of)

❌ Error handling

Feature queries allow us to condition rules based on whether particular CSS declarations are supported by the current browser

```
@supports ( declaration ) {

    /* ... */

}
```

Detect if a feature is supported

```
@supports (display: grid) {

    main {

        display: grid;

    }

}
```

Detect if all in a group of features are supported

```
@supports ( (font-kerning: normal) and

            (font-feature-settings: “kern” 1) ) {

    p {

        font-kerning: normal;

        font-feature-settings: “kern” 1;

    }

}
```

Detect if any within a group of features are supported

```
@supports ( (transform: translate(-50%, -50%)) or

            (-webkit-transform: translate(-50%, -50%)) ) {

    div {

        -webkit-transform: translate(-50%, -50%);

        transform: translate(-50%, -50%);

    }

}
```

Detect if a feature is not supported

```
@supports not (display: grid) {

    main {

        display: table;

    }

}
```

CSS Feature Queries

CSS Feature Queries allow authors to condition rules based on whether particular property declarations are supported in CSS using the @supports at rule.

IE	Edge	Firefox	Chrome	Safari	iOS Safari	Opera Mini	Chrome for Android	Android Browser	Samsung Internet
9	80	76	80	12.1	13.2			4.4	10.1
10	81	77	81	13	13.3			4.4.4	11.2
11	83	78	83	13.1	13.5	all	81	81	12.0
		79	84	14	14.0				
		80	85	TP					

✓ ✗ Partial Support

Global: 97.04% + 0% = 97.04%

Use feature queries as an enhancement

✅

```
@supports (display: grid) {
    main {
        display: grid;
    }
}
```

❌

```
@supports not (display: grid) {
    main {
        display: table;
    }
}
```

	Supports Feature Queries ✅	**Does Not** Support Feature Queries ❌
Supports CSS Feature ✅		
Does Not Support CSS Feature ❌		

	Supports Feature Queries ✅	**Does Not** Support Feature Queries ❌
Supports CSS Feature ✅		
Does Not Support CSS Feature ❌		

Browser doesn’t support FQs and does not support CSS grid ✅

```
@supports (display: grid) {

    main {

        display: grid;

    }

}
```

	Supports Feature Queries ✅	**Does Not** Support Feature Queries ❌
Supports CSS Feature ✅	😊	😩
Does Not Support CSS Feature ❌	😊	😊

Browser doesn’t support FQs
but does support CSS grid ❌

```
@supports (display: grid) {

    main {

        display: grid;

    }

}
```

WHAT WILL USERS LOOK LIKE IN THE FUTURE ?


MORE PEOPLE

In 1999, just 4% of the global population were online

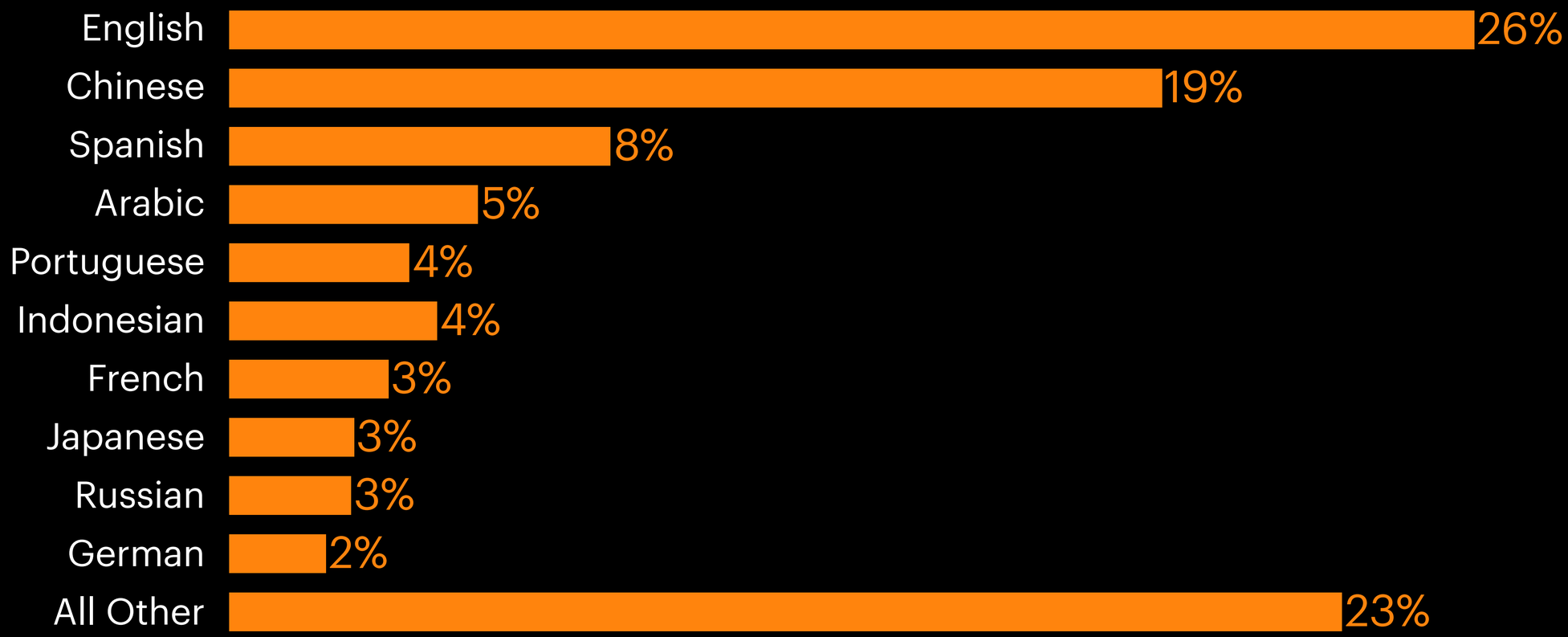
http://www.itu.int/ITU-D/ict/statistics/ict

In 1999, >96% of internet users were from the English-speaking world

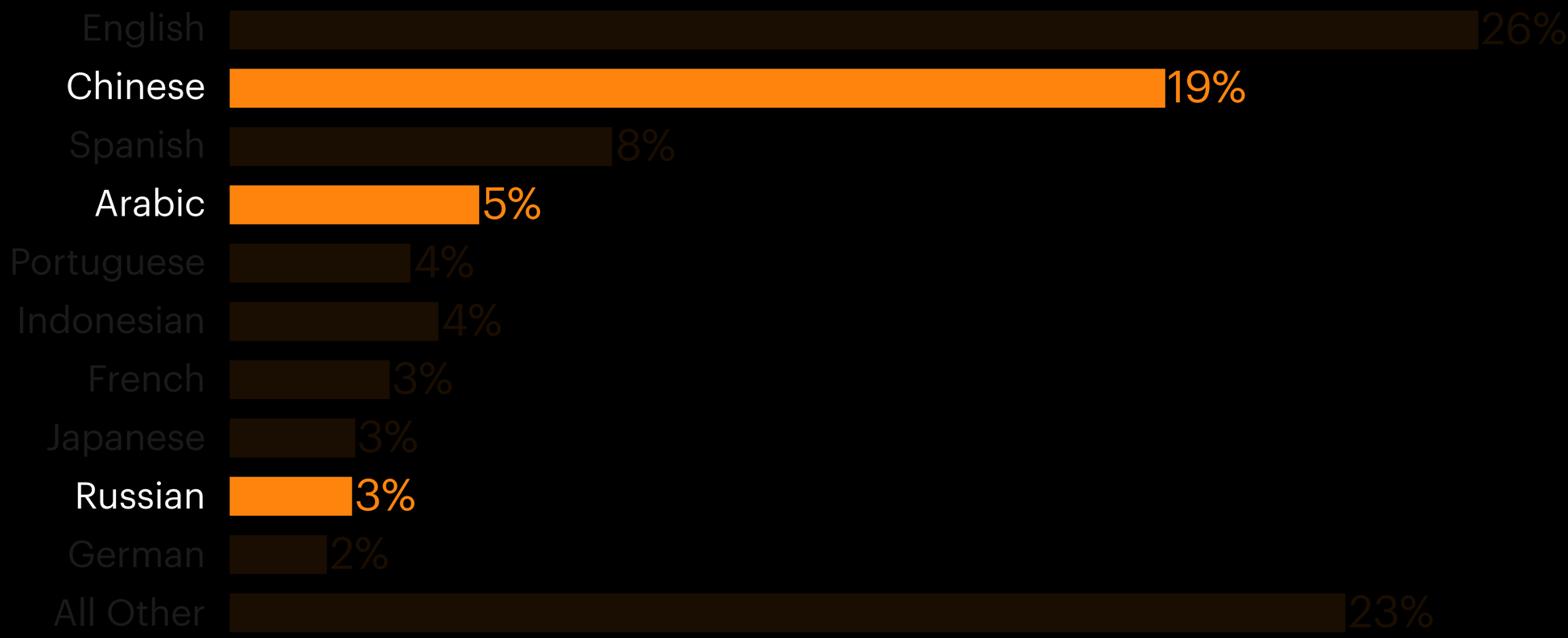
http://www.itu.int/ITU-D/ict/statistics/ict

In 2019, over 53% of the global population are online

https://www.statista.com/statistics/273018/number-of-internet-users-worldwide



https://www.internetworldstats.com/stats7.htm



https://www.internetworldstats.com/stats7.htm

Русский использует совершенно другой алфавит

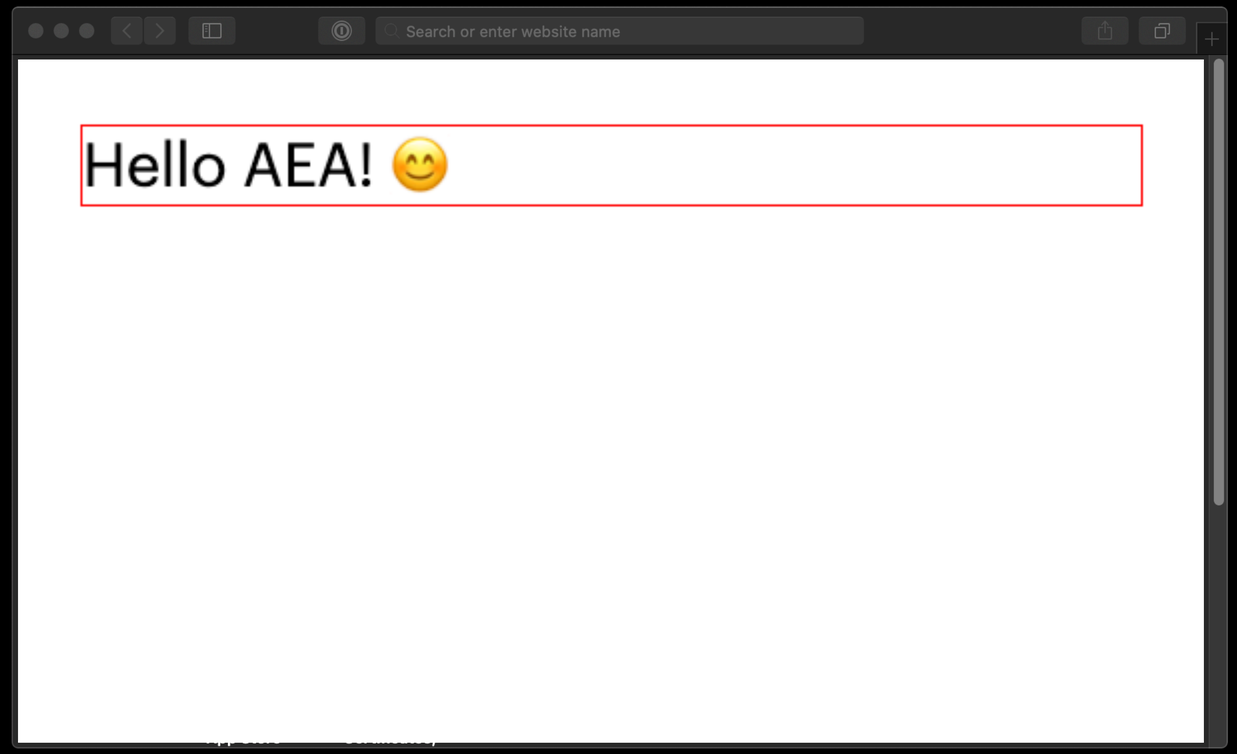
تتم قراءة اللغة العربية من اليمين إلى اليسار

中文可以从右到左
和从上到下阅读


INTERNATIONALISATION

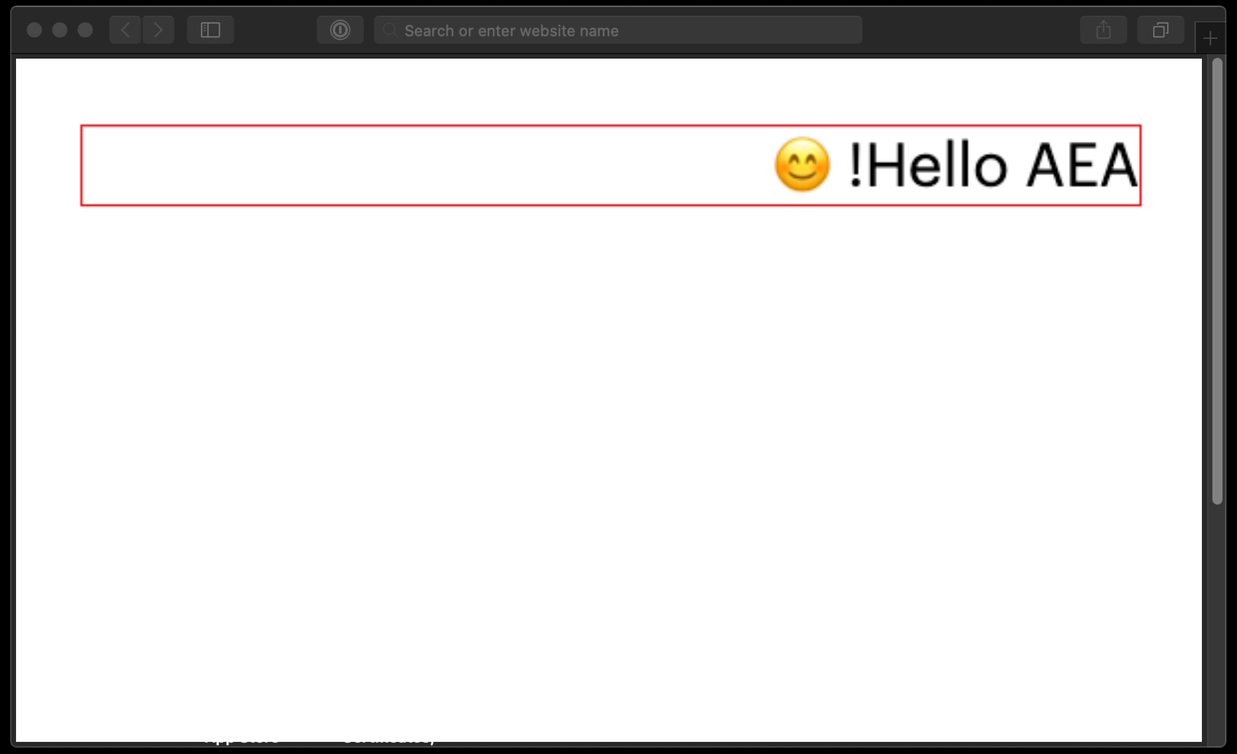
direction & writing-mode & text-orientation & unicode-bidi

```
<div>Hello AEA!</div>
```



Set direction and alignment

```
div {

    direction: rtl;

}
```



Set flow

```
div {

    writing-mode: vertical-lr;

}
```

```
div {
    direction: rtl;
    writing-mode: vertical-lr;
}
```

Set orientation

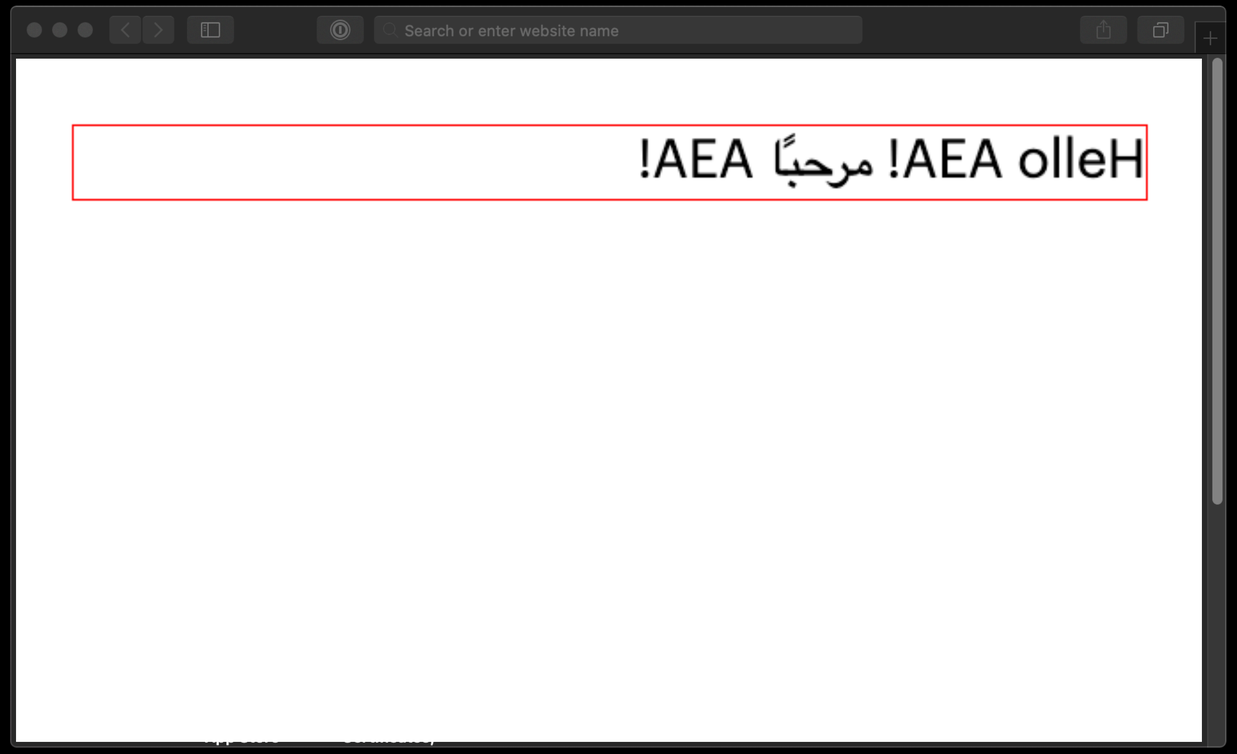
```
div {

    direction: rtl;

    writing-mode: vertical-lr;

    text-orientation: sideways;

}
```

Handle multiple languages and directions

```
div {

    direction: rtl;

}
```

Handle multiple languages and directions

```
div {

    direction: rtl;

    unicode-bidi: bidi-override;

}
```



Use logical CSS properties

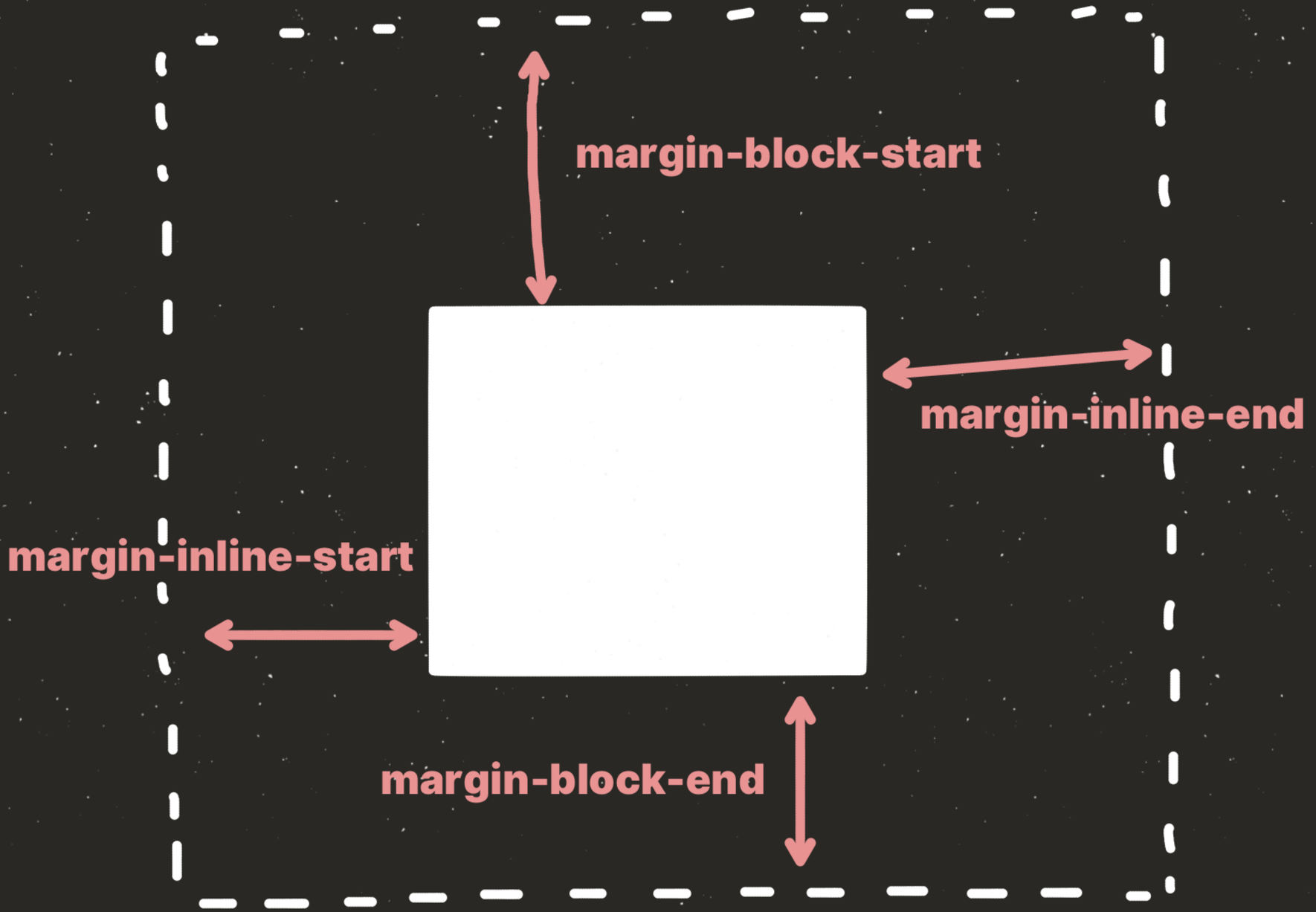
CSS has, so far, largely been written for English alone ☹️

Left usually doesn’t mean left

```
p {
    margin-left: 10px;
}
```

Left usually doesn’t mean left

```
[lang=“ar”] {

    direction: rtl;

}

[lang=“ar”] p {

    margin-right: 10px;

}
```


margin-block-start
margin-inline-end
margin-inline-start
margin-block-end

padding
border
margin

inline
block

start
end

~~left~~ inline-start

```
p {
    margin-inline-start: 10px;
}
```

This is a test of CSS logical properties

هذا اختبار للخصائص المنطقية لـ CSS

Use left when you actually mean left

Use logical naming conventions

❌

```
.nav-right { ... }
```

✅

```
.nav-aside { ... }

.nav-start { ... }

.nav-end { ... }
```

MORE USERS = MORE VARIATION IN USERS

We can’t assume one type of user

Image from https://www.maltron.com/store/p25/Maltron_Expanded_Keyboard_-_US_English.html

Image from https://theweco.com/motor-skill-related-disabilities-navigating-and-designing-websites

Image from https://axesslab.com/switches

Today, 15% of the global population need to use an assistive product

https://www.who.int/news-room/fact-sheets/detail/assistive-technology

Image from https://design.google/library/designers-guide-accessibility-research

HTML is by default accessible.

It's our jobs, as developers, to not f😮ck that up.

— @estellevw

Only use CSS for styling

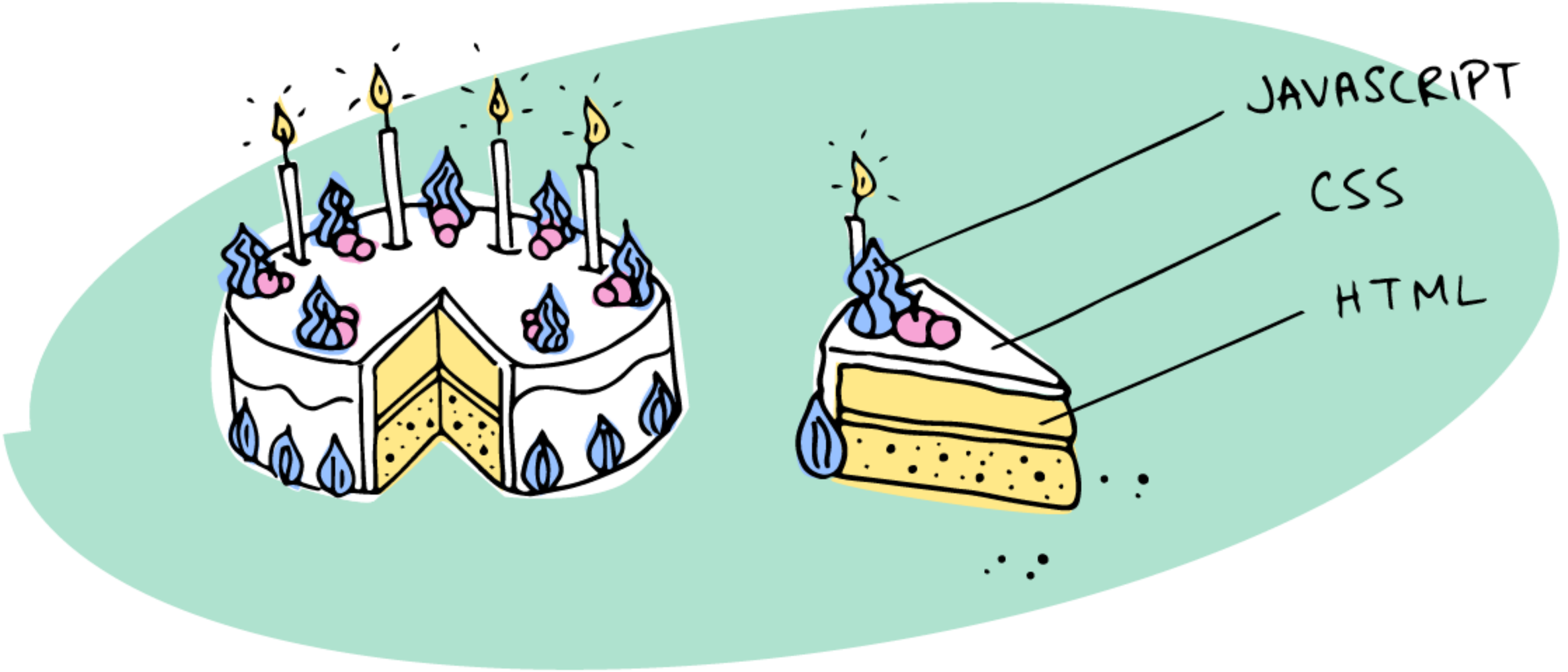


Image from https://www.shopify.com/partners/blog/what-is-progressive-enhancement-and-why-should-you-care

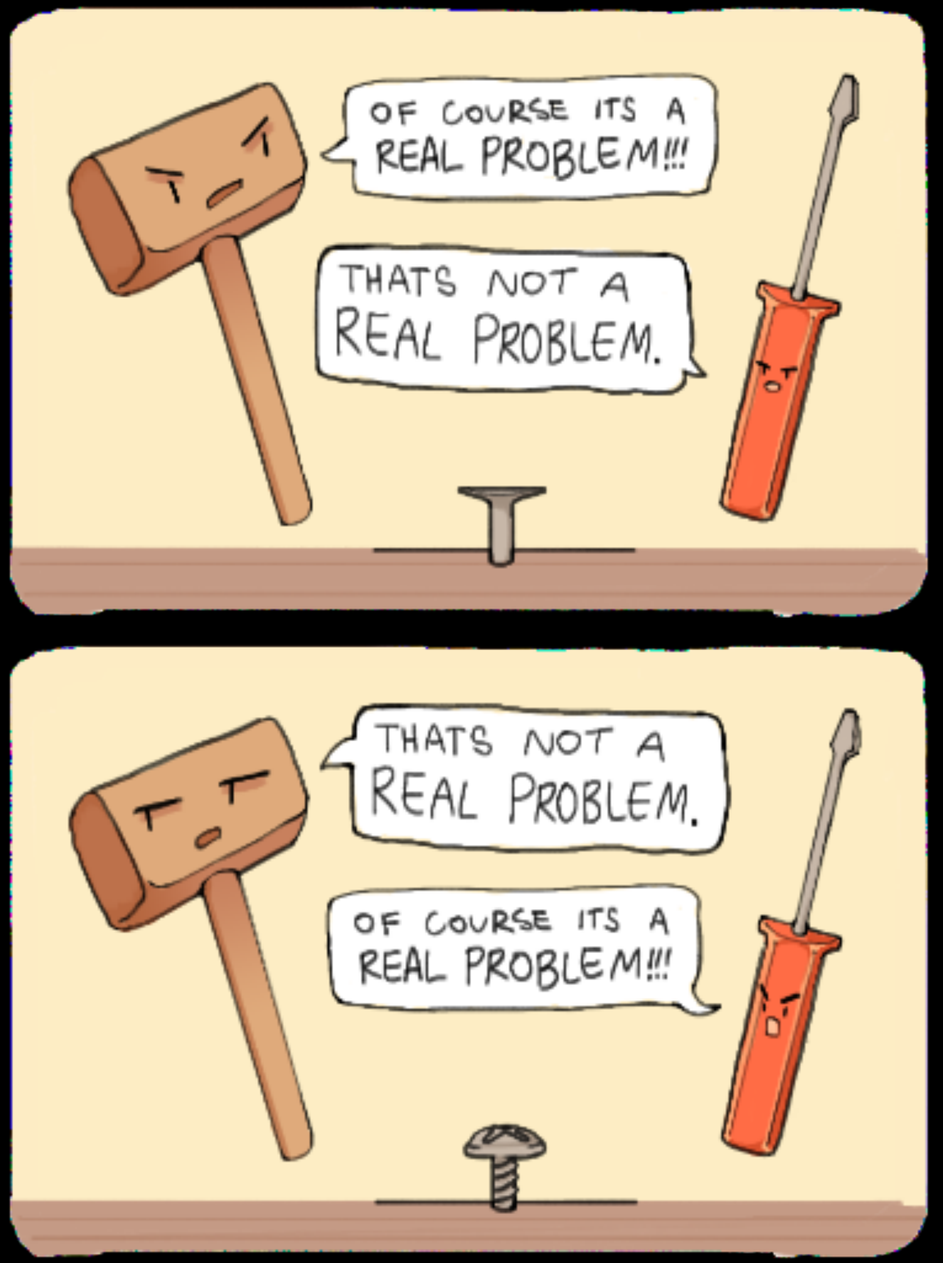
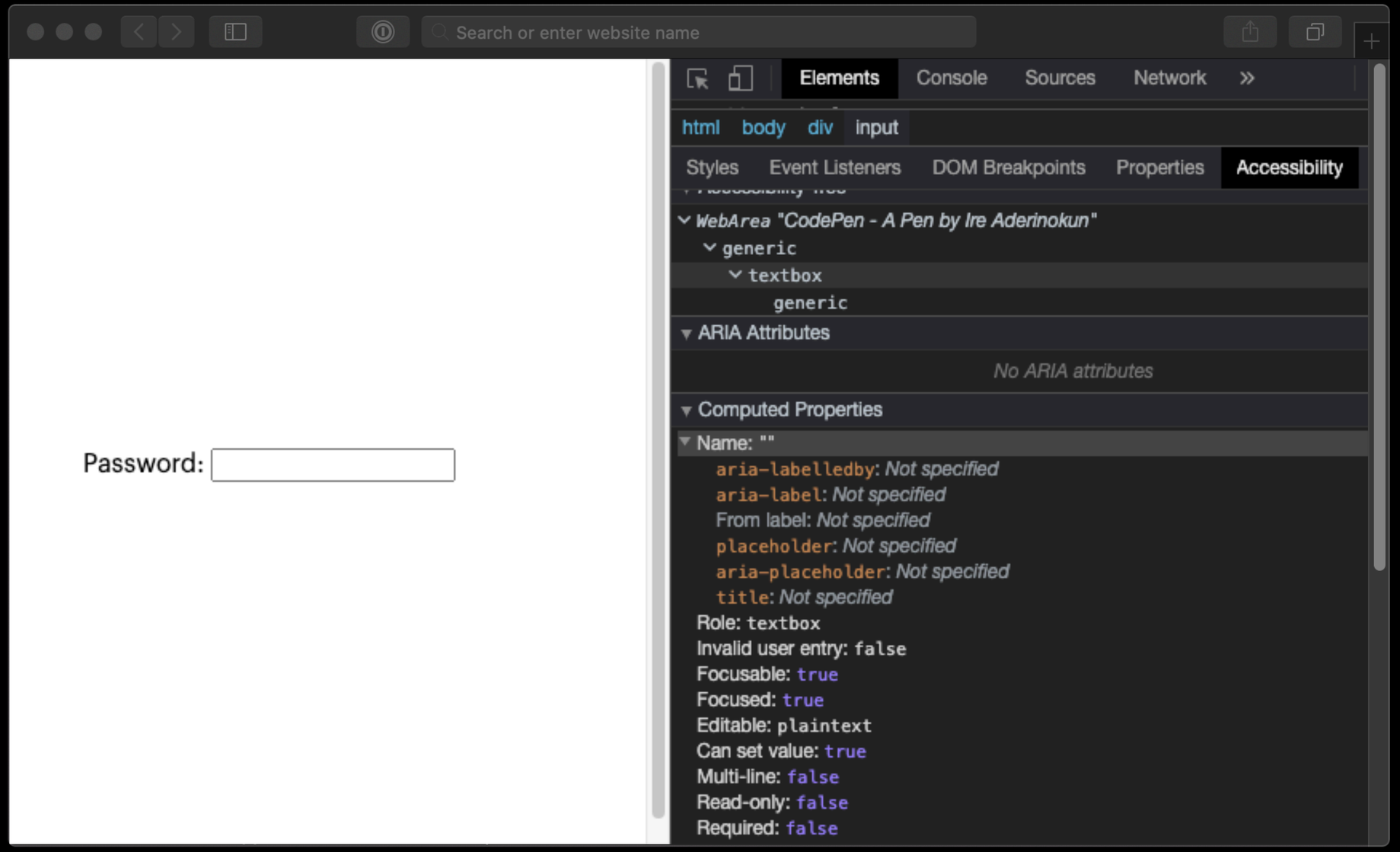


Image from https://www.dr4ward.com/dr4ward/2015/09/what-is-solution-aversion-and-the-law-of-the-hammer-how-does-it-impact-change-comic-.html

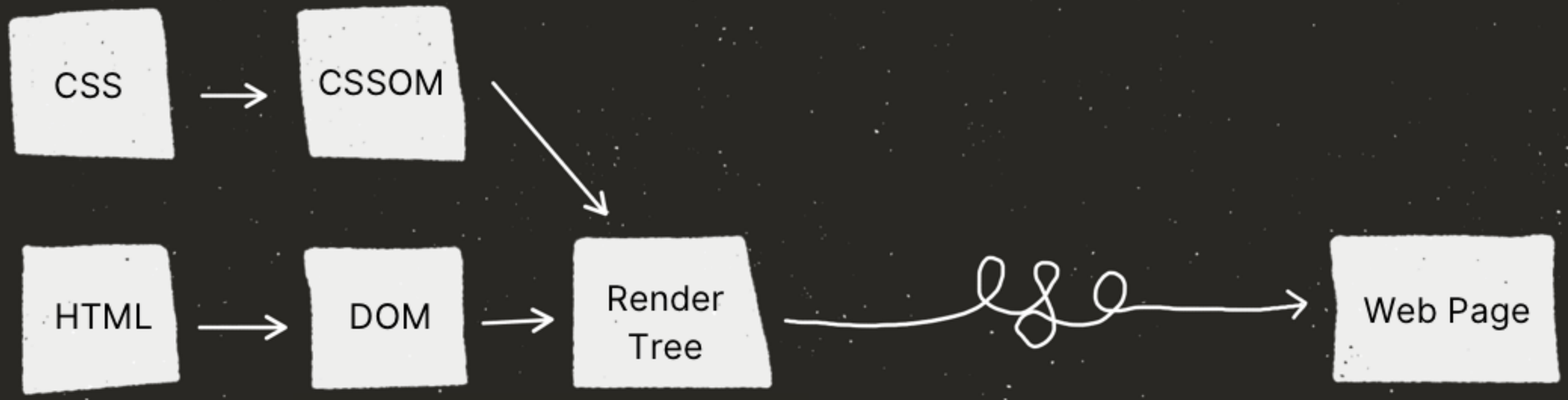
Don’t use CSS for content

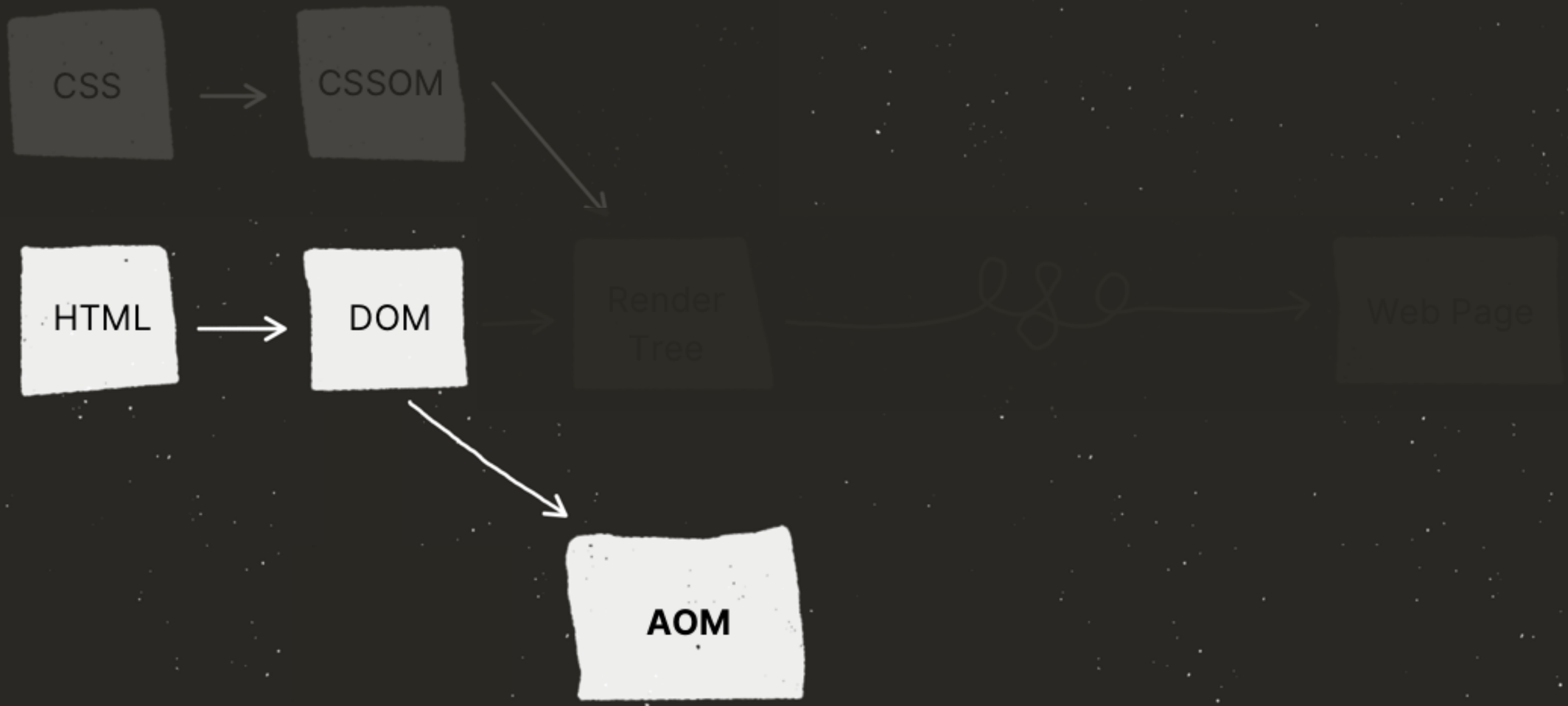
```
<div>
    <input type=“password”>
</div>
```

```
div::before {
    content: “Password: ”;
}
```


Search or enter website name
Password:
Elements
Console
Sources
Network
html
body
div
input
Styles
Event Listeners
DOM Breakpoints
Properties
Accessibility
WebArea "CodePen - A Pen by Ire Aderinokun"
generic
textbox
generic
ARIA Attributes
No ARIA attributes
Computed Properties
Name: ""
aria-labelledby: Not specified
aria-label: Not specified
From label: Not specified
placeholder: Not specified
aria-placeholder: Not specified
title: Not specified
Role: textbox
Invalid user entry: false
Focusable: true
Focused: true
Editable: plaintext
Can set value: true
Multi-line: false
Read-only: false
Required: false

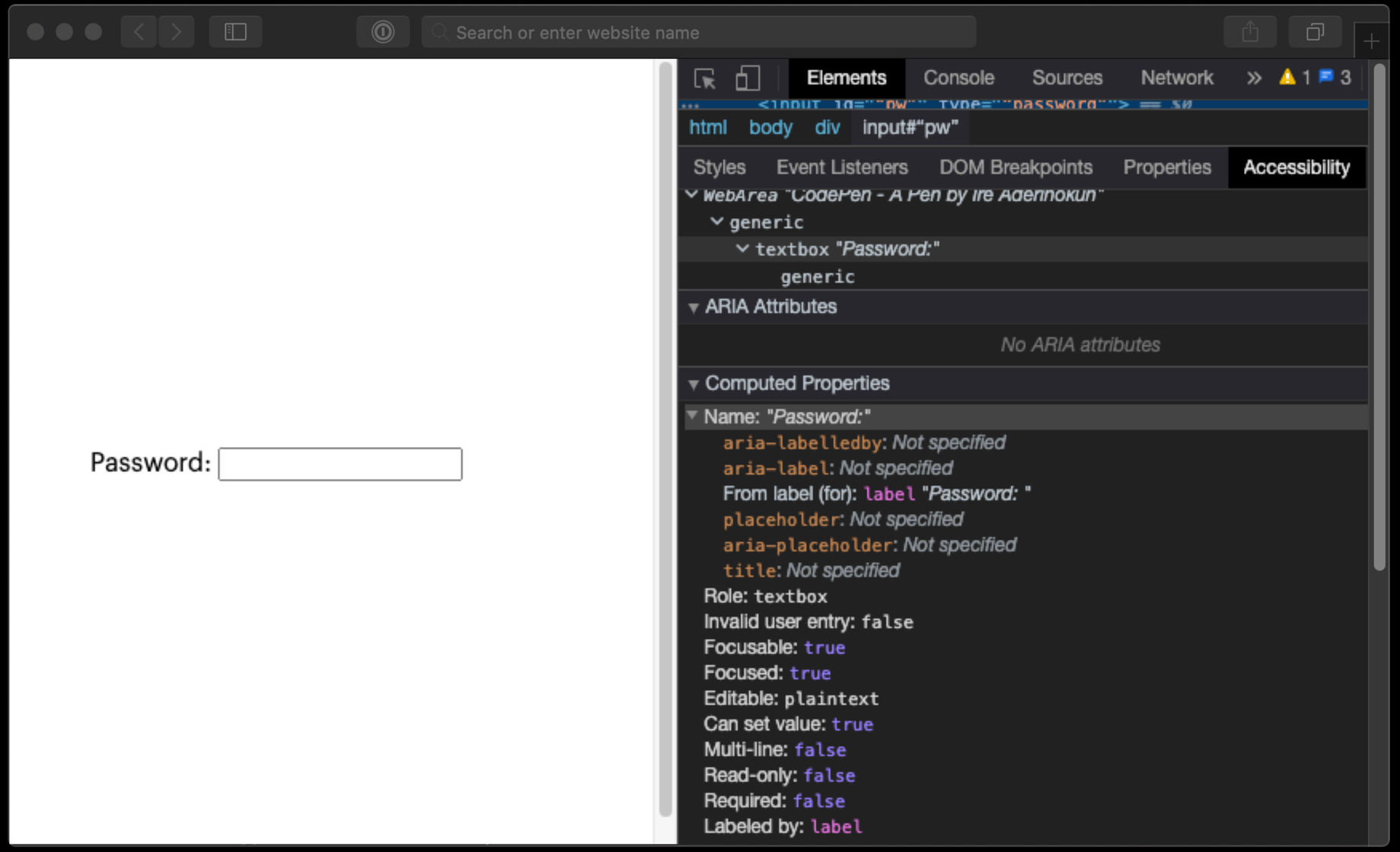
CSS is not part of the accessibility tree 🌳


CSS
CSSOM
HTML
DOM
Render Tree
Web Page


CSS
CSSOM
HTML
DOM
Render Tree
Web Page
AOM

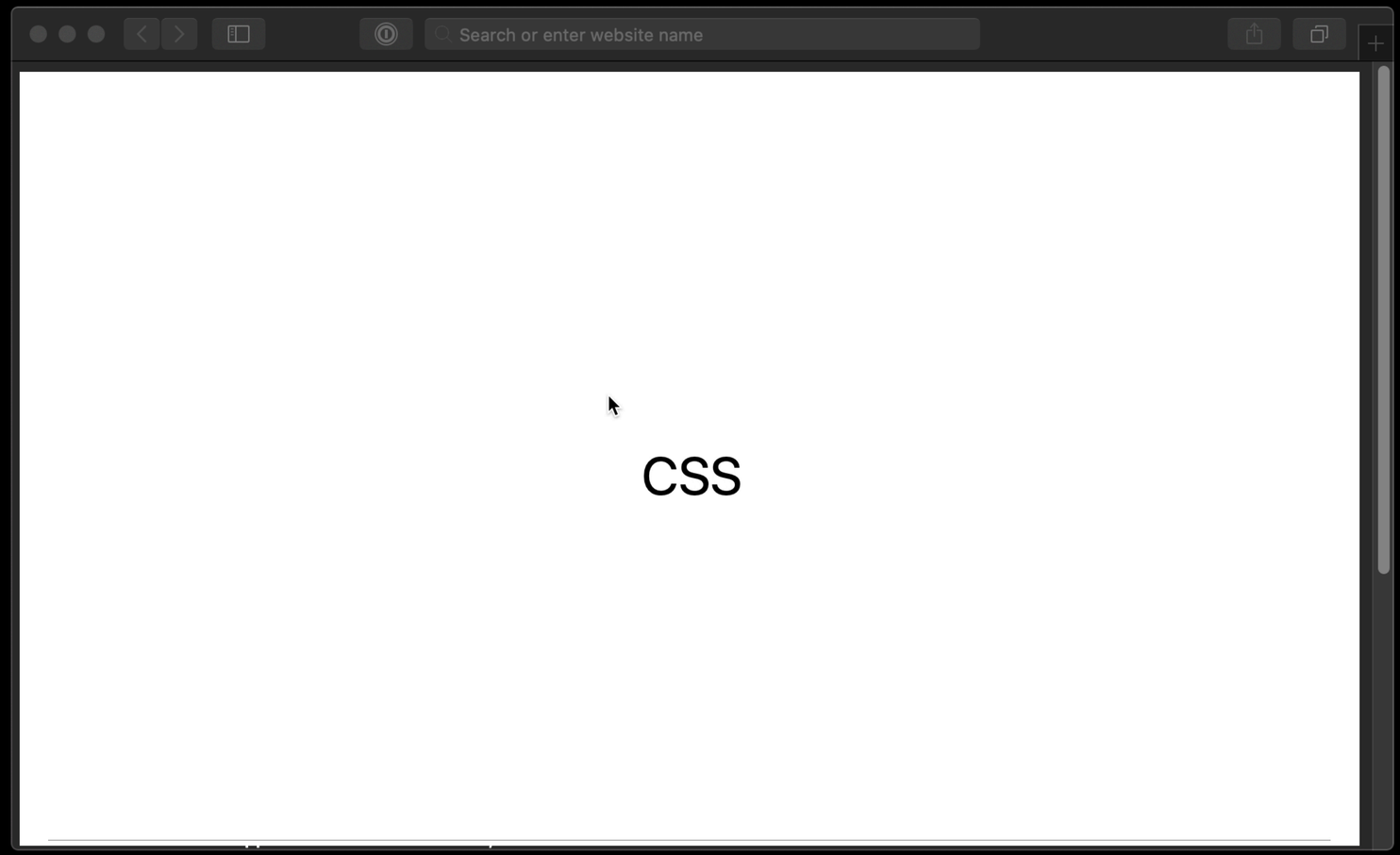
✅

```
<div>
    <label for=“pw”>Password</label>
    <input id=“pw” type=“password”>
</div>
```


Search or enter website name
Password:
Elements
Console
Sources
Network
1
3
html
body
div
input#"pw"
Styles
Event Listeners
DOM Breakpoints
Properties
Accessibility
WebArea "CodePen - A Pen by Ire Aderinokun"
generic
textbox "Password:"
generic
ARIA Attributes
No ARIA attributes
Computed Properties
Name: "Password:"
aria-labelledby: Not specified
aria-label: Not specified
From label (for): label "Password: "
placeholder: Not specified
aria-placeholder: Not specified
title: Not specified
Role: textbox
Invalid user entry: false
Focusable: true
Focused: true
Editable: plaintext
Can set value: true
Multi-line: false
Read-only: false
Required: false
Labeled by: label

Don’t use CSS for functionality

Trying to make the <abbr> element accessible with only CSS



https://bitsofco.de/making-abbr-work-for-touchscreen-keyboard-mouse

Issues

- ❌ Used tabindex attribute on a non-interactive element
- ❌ Couldn't dismiss the tooltip on mobile devices
- ❌ Duplication of content for screenreaders

Solved with JS

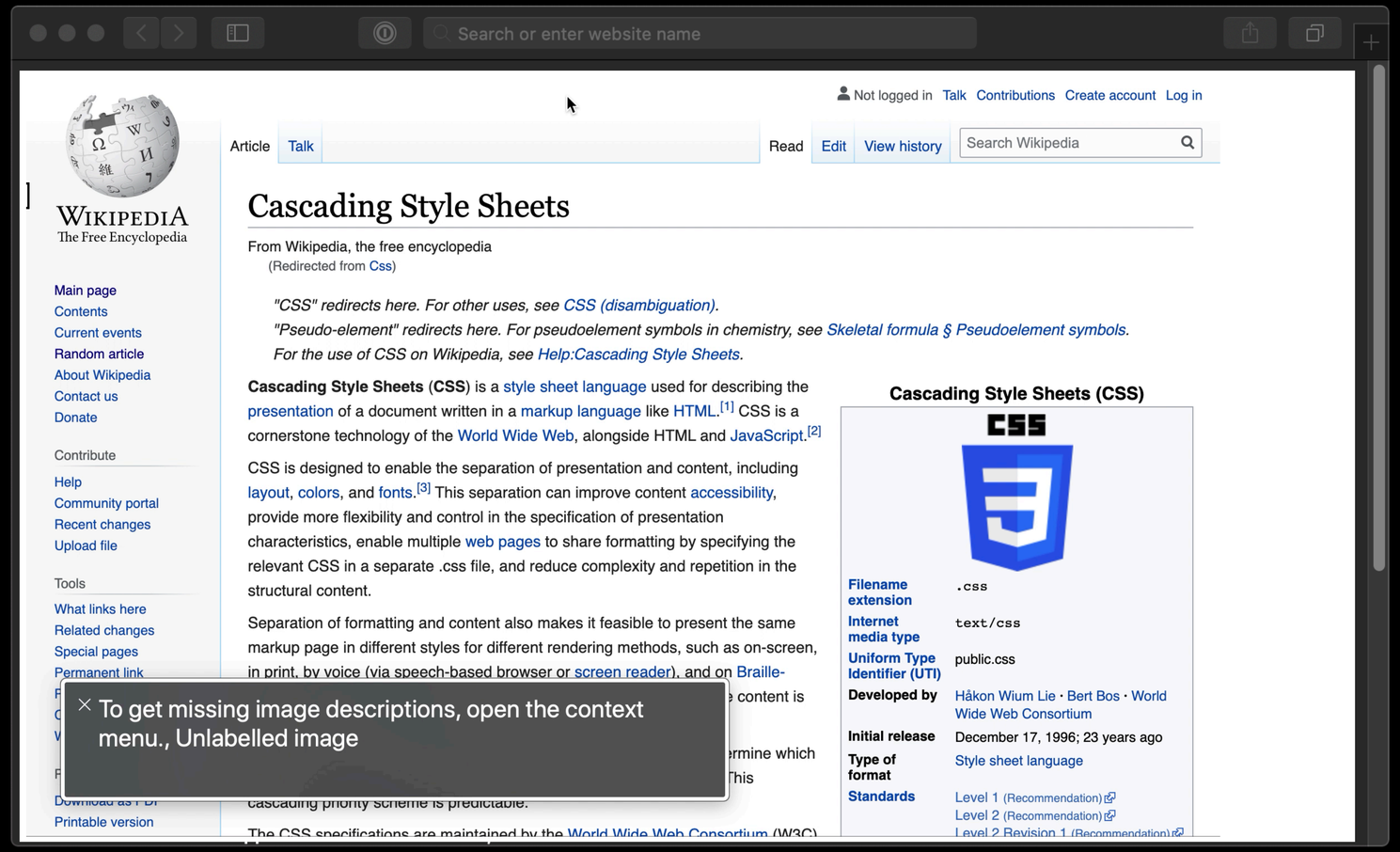
Change, don’t undo, the default accessible styles

✅ Accessible to various input devices

✅ Hover, focus, and active states

✅ Semantically meaningful

✅ Using assistive technology, interactive elements can be easily navigated to



We can still change the default styles

❌

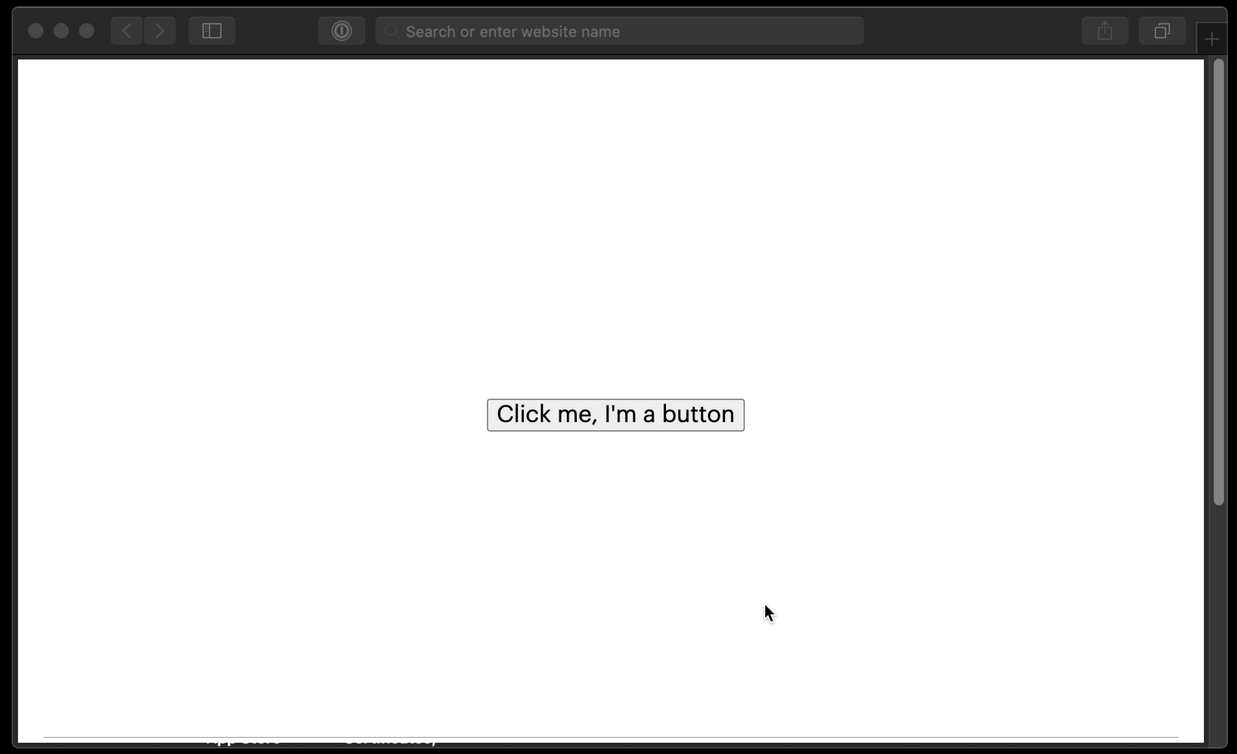
```
:focus {
    outline: none;
}
```

✅

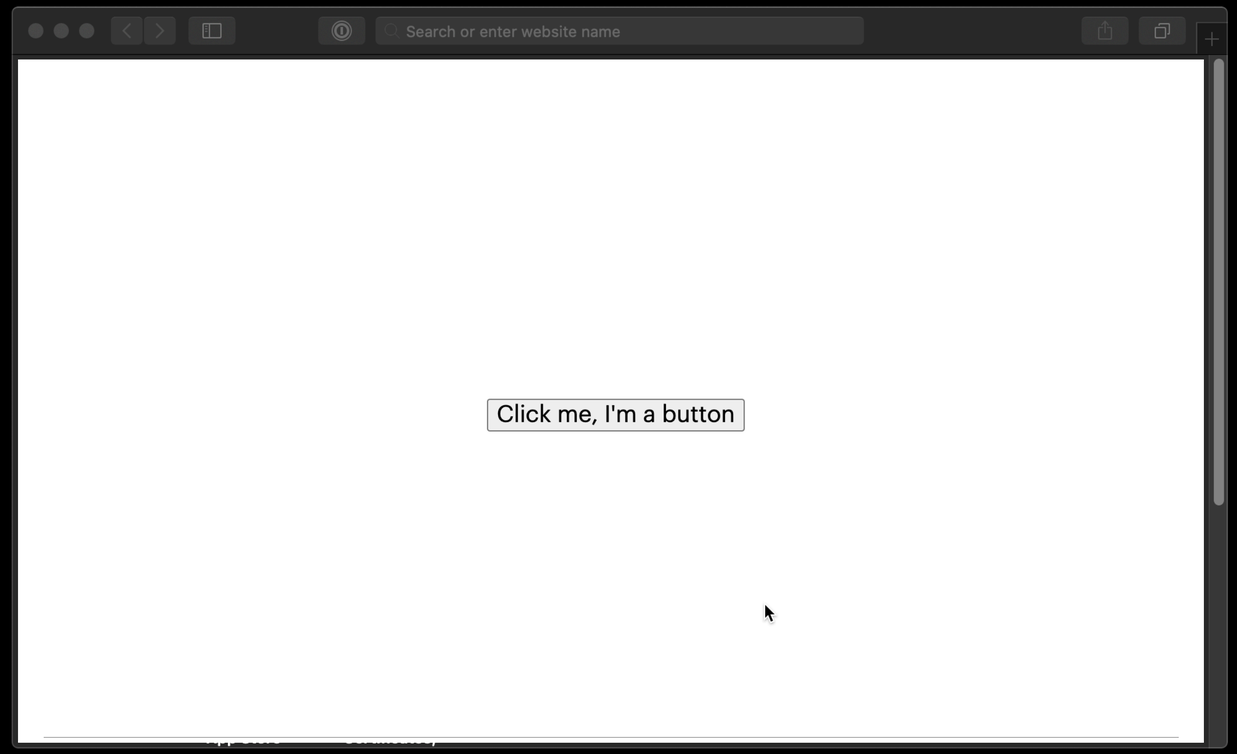
```
:focus {
    outline: none;
    background-color: darksalmon;
}
```

The :focus-visible pseudo-class applies focus styles only in cases where the browser determines focus needs to be visible

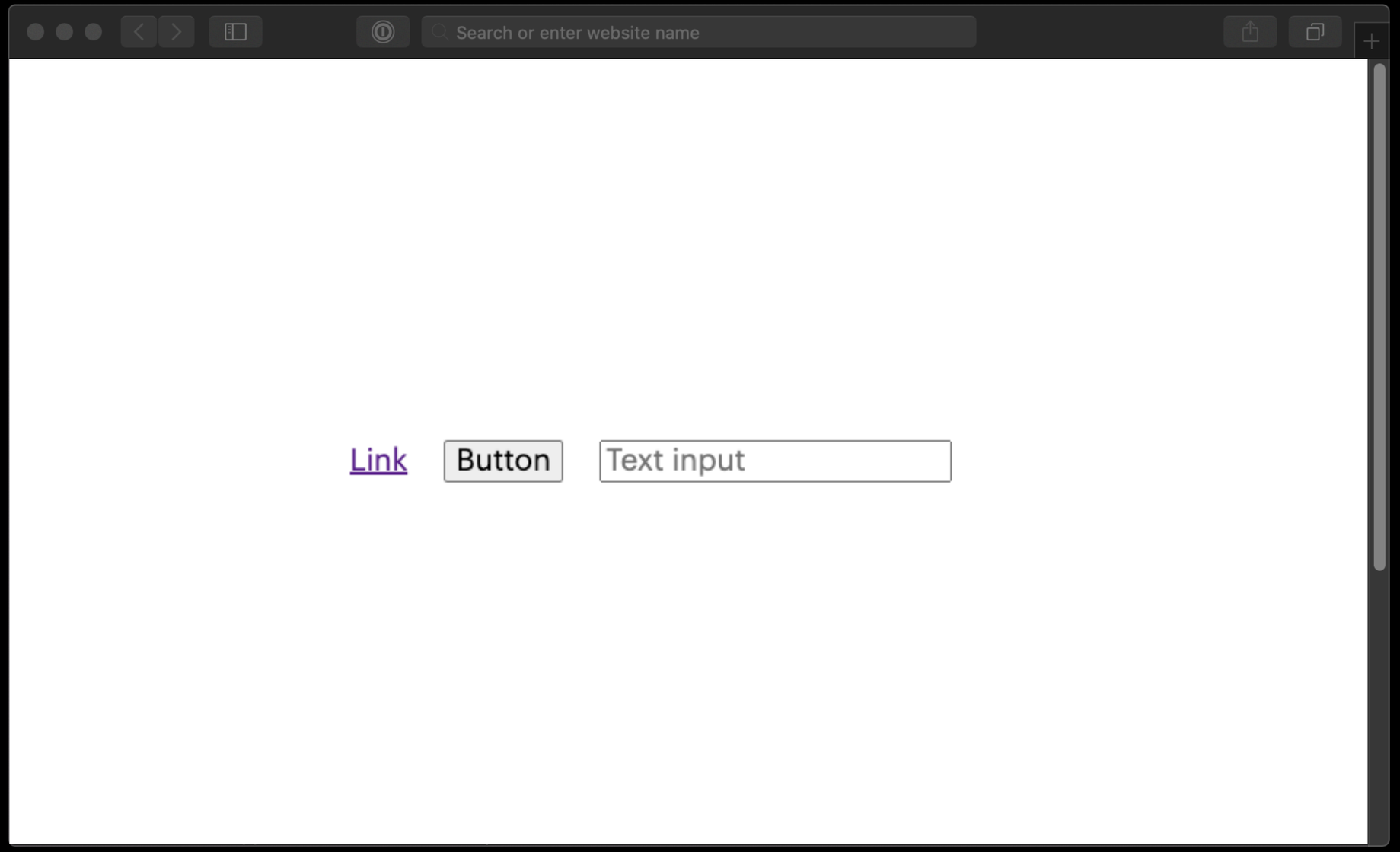
```
button:focus:not(:focus-visible) {
  outline: none;
}

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



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

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



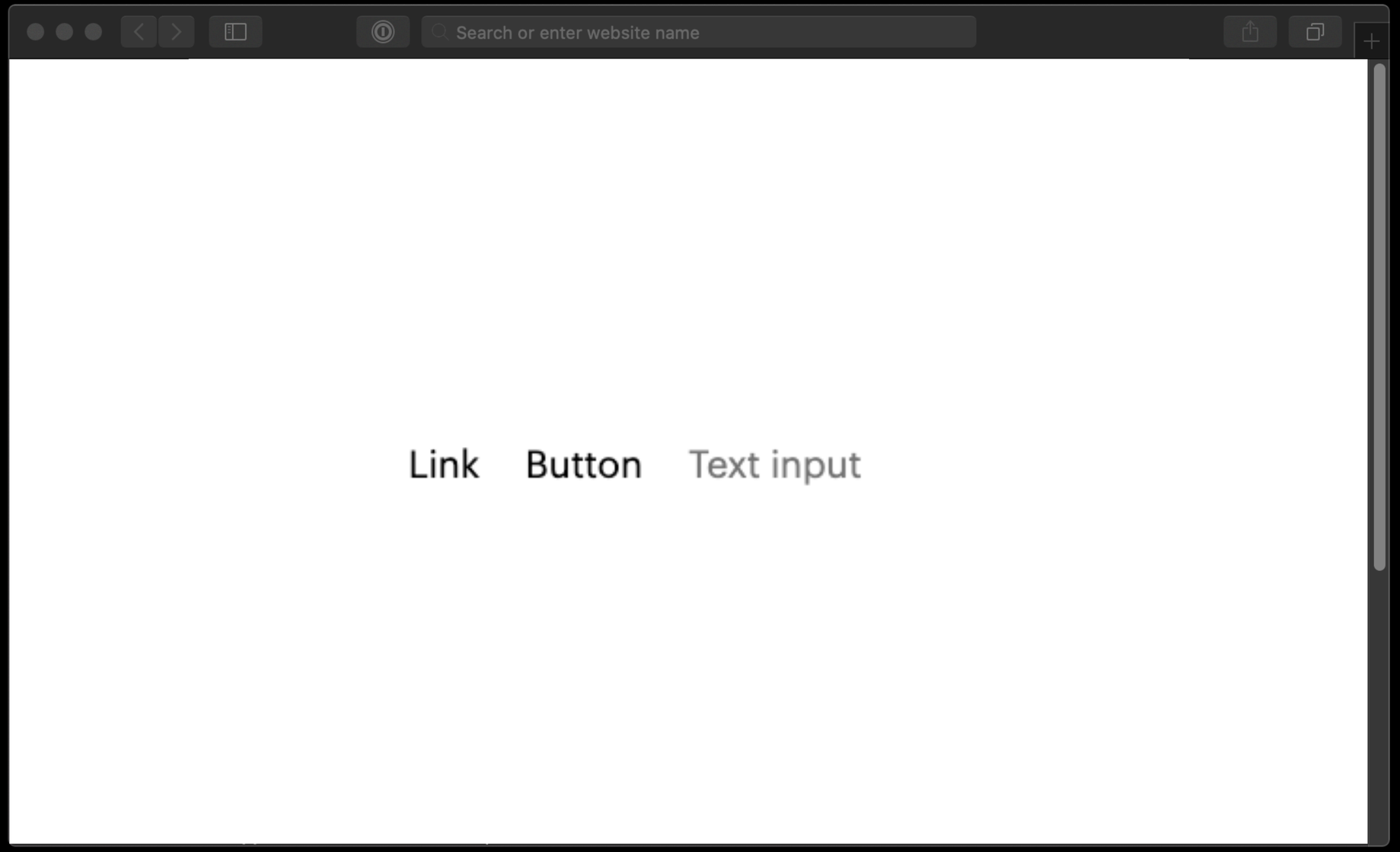
Follow styling conventions


Search or enter website name
Link
Button
Text input

One material should not be used as a substitute for another, otherwise the end result is deceptive

— Jeremy Keith

https://resilientwebdesign.com


Search or enter website name
Link
Button
Text input

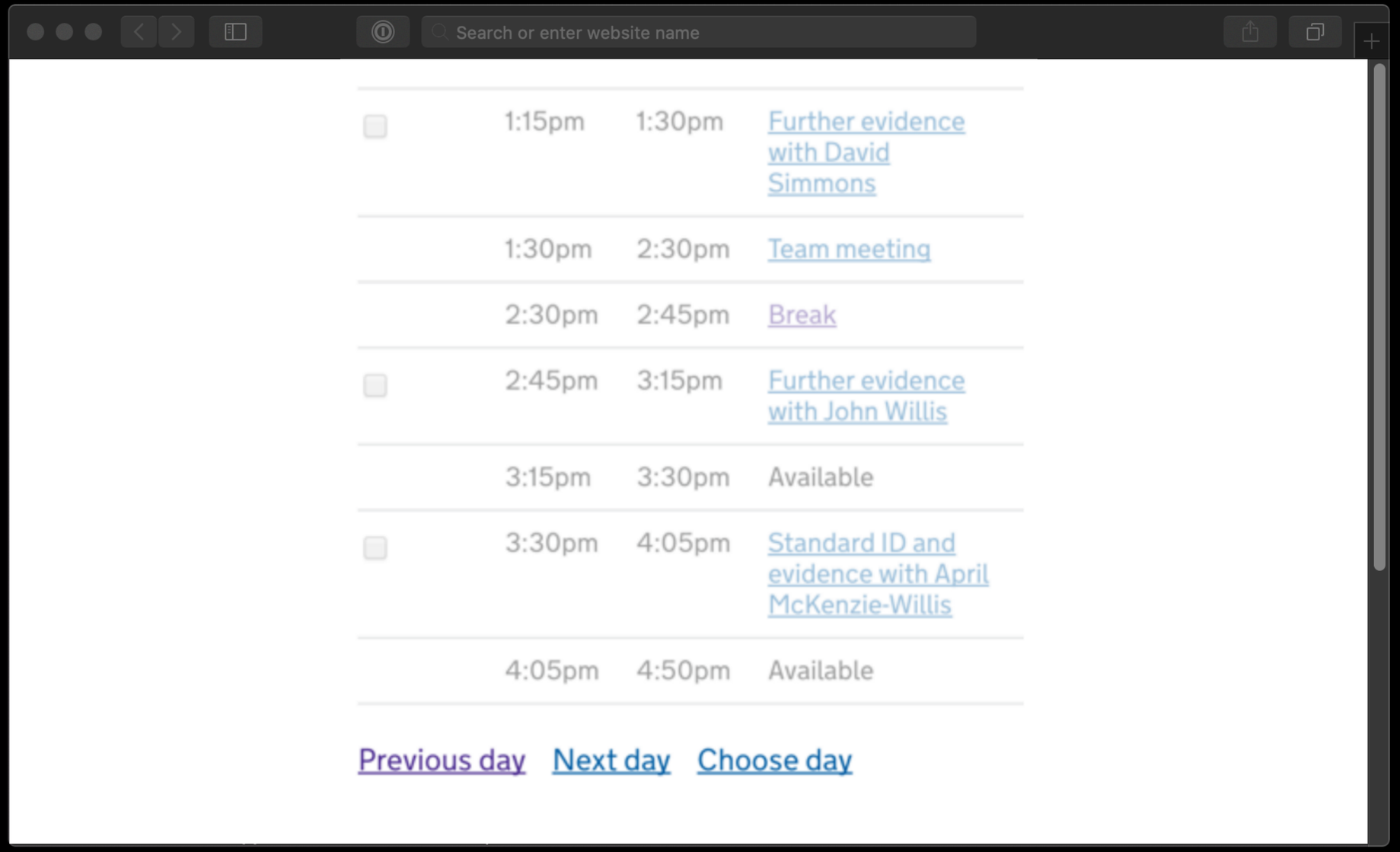
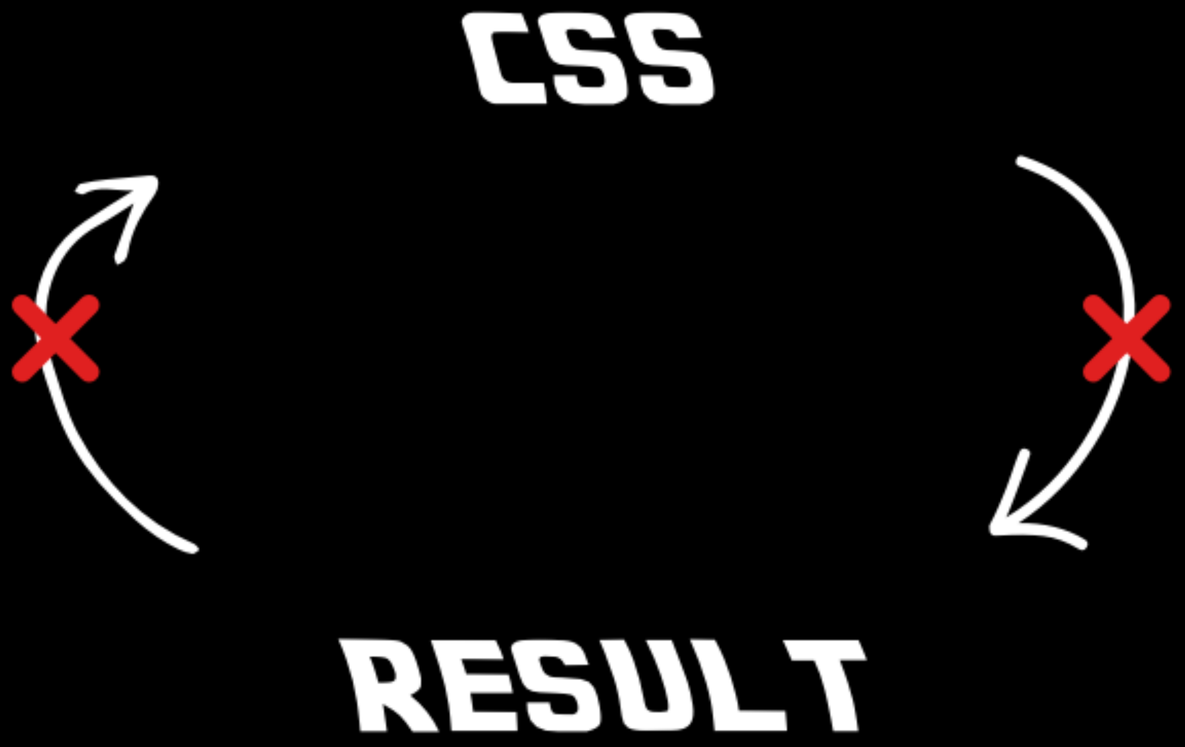
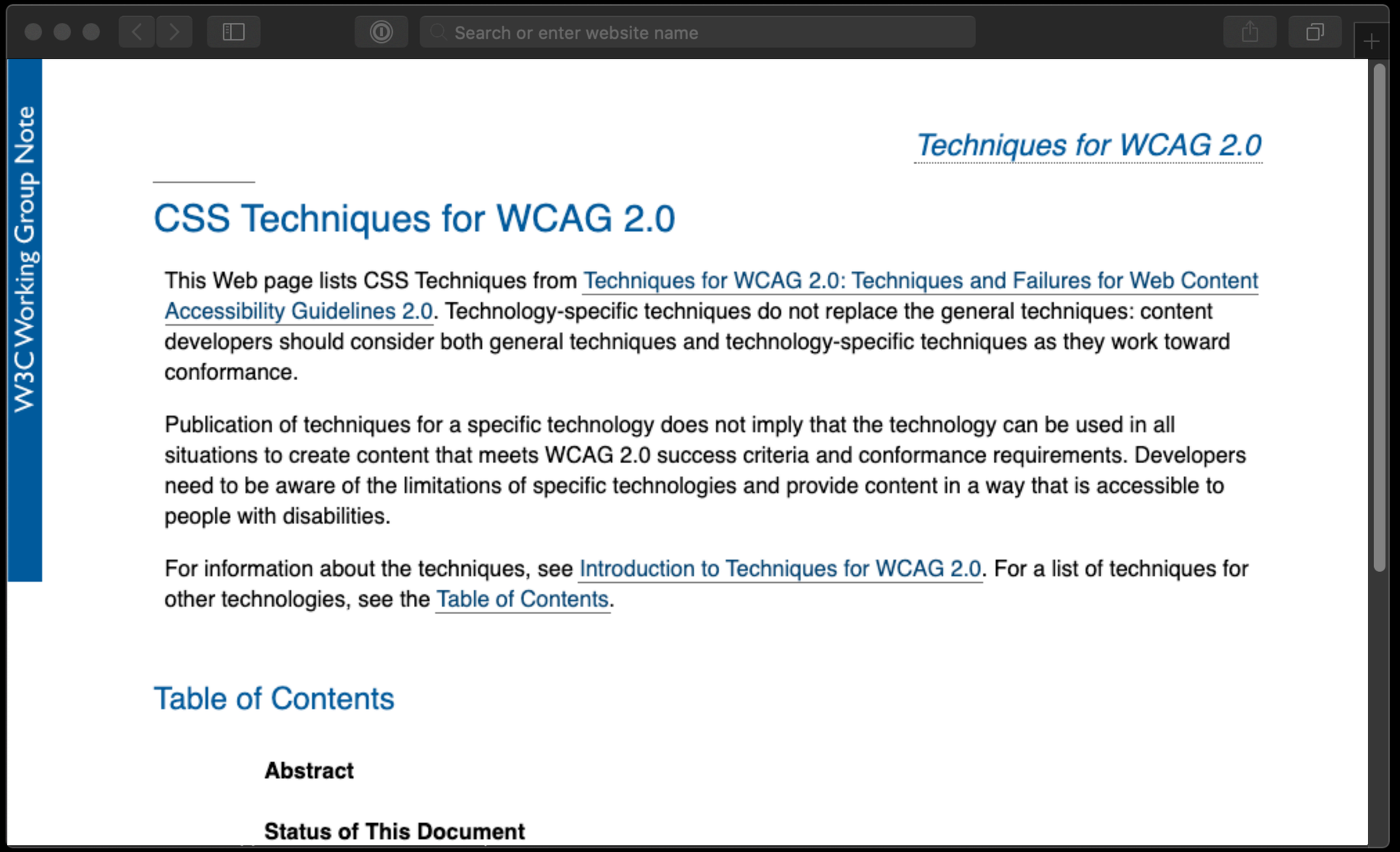


Image from https://medium.com/simple-human/but-sometimes-links-look-like-buttons-and-buttons-look-like-links-9b371c57b3d2

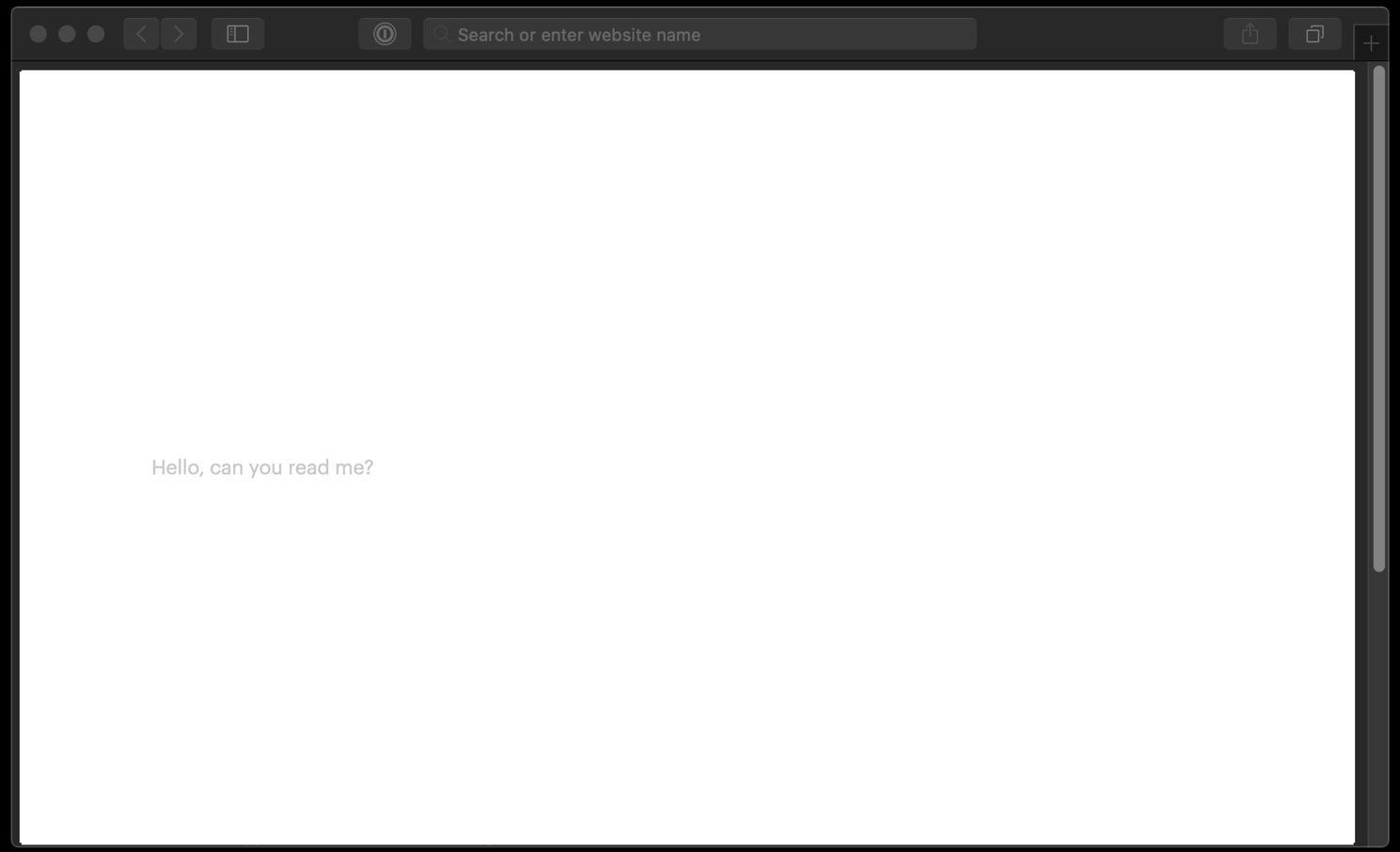
Be proactive about compliance


CSS
RESULT

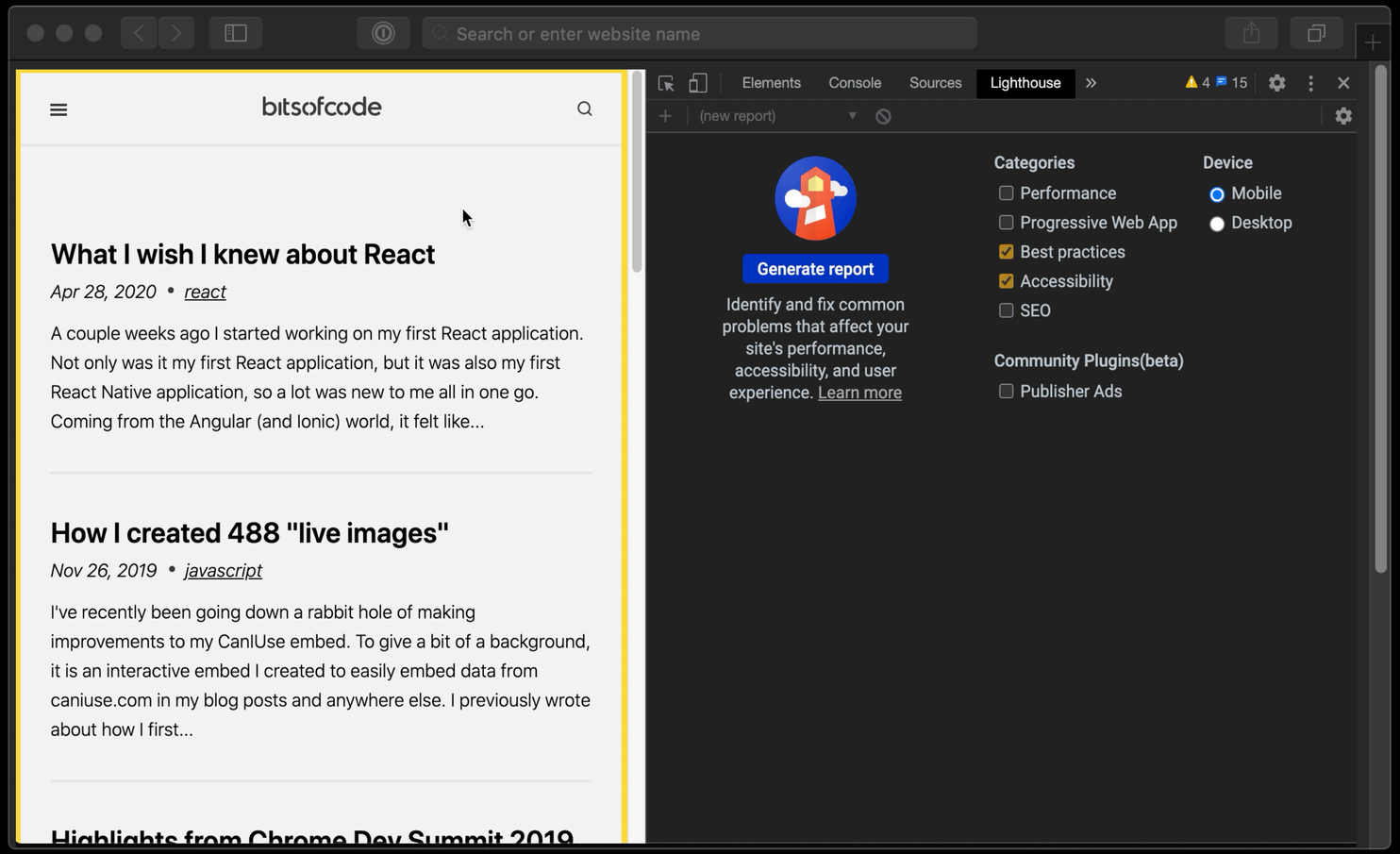


https://www.w3.org/TR/WCAG20-TECHS/css

Firefox accessibility inspector

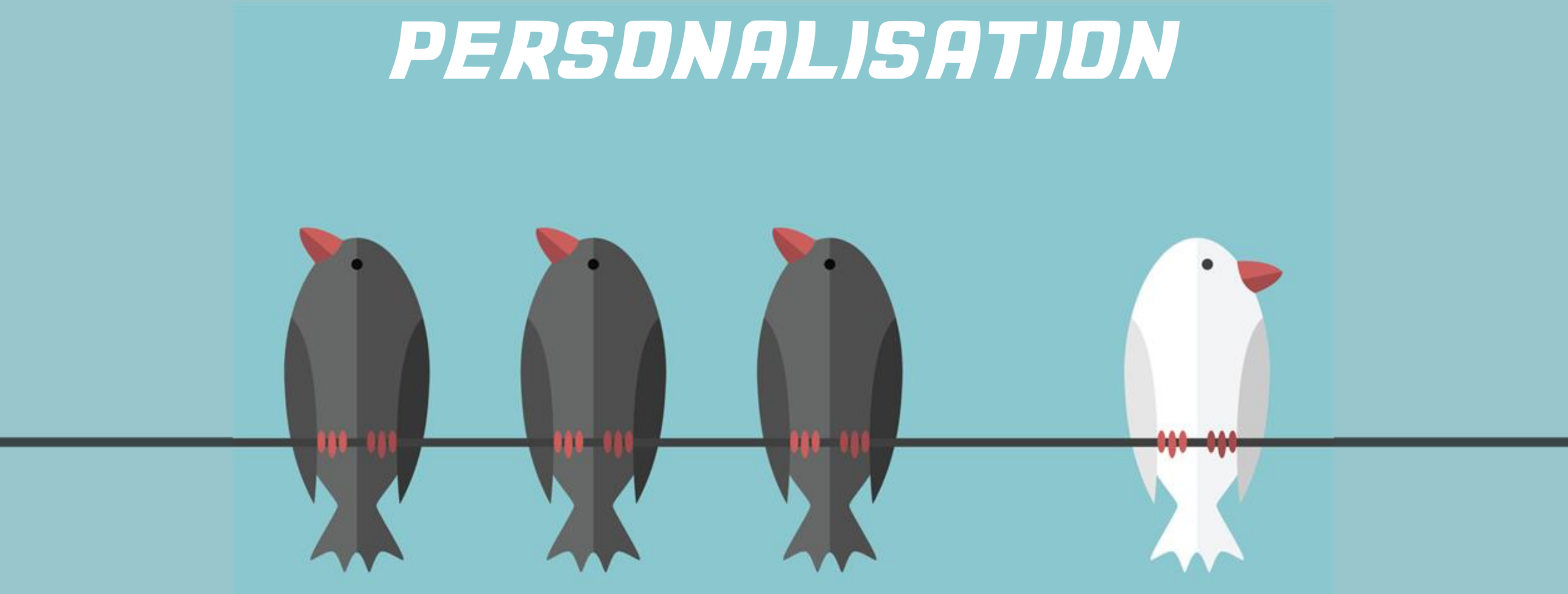


Lighthouse accessibility report



CSS is subjective, so tools can only tell you so much

Image from https://www.citmagazine.com/article/1526238/why-personalisation-key-measuring-events-success



Write adaptive CSS

Adapt to **colour** preferences

```
body { background-color: white; color: black; }


@media (prefers-color-scheme: dark) {
    body { background-color: black; color: white; }
}
```

Adapt to **animation** preferences

```
element { animation: 3s infinite alternate slideIn; }


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

    .element { animation: none; }

}
```

Adapt to **data** preferences

```
.element { background-image: url(“image-1800w.png”); }


@media (prefers-reduced-data: reduce) {

    .element { background-image: url(“image-600w.jpg”); }

}
```


CSS at-rule: @media: prefers-reduced-data media feature
Usage % of all users
Global
0%
Current aligned
Usage relative
Date relative
IE
Edge
Firefox
Chrome
Safari
Opera
iOS Safari
Opera Mini
Android Browser
Opera Mobile
Chrome for Android
Firefox for Android
UC fo
6-10
12-8
2-7
4-81
10-68
3.2-13.3
12-12
11
83
78
83
13.1
46
81
68
79-80
TP
4.0
5-
Progressive enhancement

Use rems over pixels for sizes

REM represents the font-size of the root element (typically <html>). When used on the font-size on this root element, it represents its initial value.

— MDN Web Docs

https://developer.mozilla.org/en-US/docs/Web/CSS/length#rem

Using rems allows a website to automatically adapt to user preferences


Search or enter website name
I'm 16px
I'm 1rem

FUTURE-PROOF CSS 🚀


Future-proofing == past-proofing

Progressive enhancement

✅ Start with sensible HTML

✅ Use the cascade

✅ Write mobile-first CSS

✅ Don’t be afraid to use the “old thing”

✅ Use feature queries (as an enhancement)

Internationalisation

✅ Use direction, writing-mode, text-orientation, & unicode-bidi

✅ Use logical CSS properties

✅ Use logical naming conventions

Accessibility

✅ Only use CSS for styling

✅ Change, don’t undo, the default accessible styles

✅ Follow styling conventions

✅ Be proactive about compliance

Personalisation

✅ Write adaptive CSS

✅ Use rems over pixels for sizes

Thank you!

Ire Aderinokun 🇳🇬

COO & VP Engineering of BuyCoins

Google Web Expert & Cloudinary Media Expert

ireaderinokun.com

bitsofco.de

@ireaderinokun