


# Pragmatic Accessibility.

A practical guide to build **inclusive** web apps.

by **blurbyte**



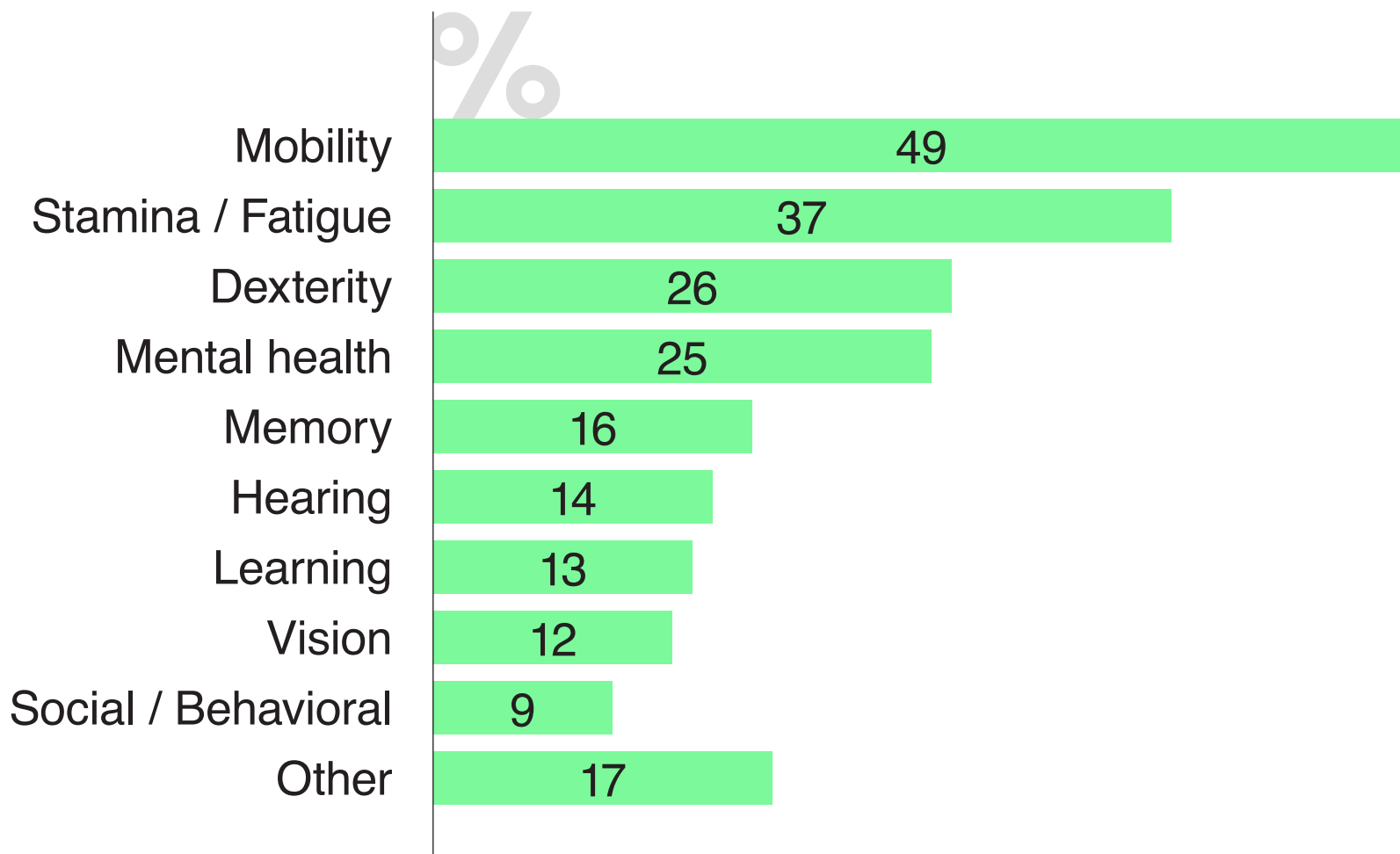
**About 1 in 5 Americans have some kind of disability, and 1 in 10 have a severe disability.<sup>1</sup>**

**21% of people living in the UK reported a disability.<sup>2</sup>**

<sup>1</sup> Disability Status: Census 2019 Brief, US

<sup>2</sup> Family Resources Survey 2017/18, UK

# Impairment types

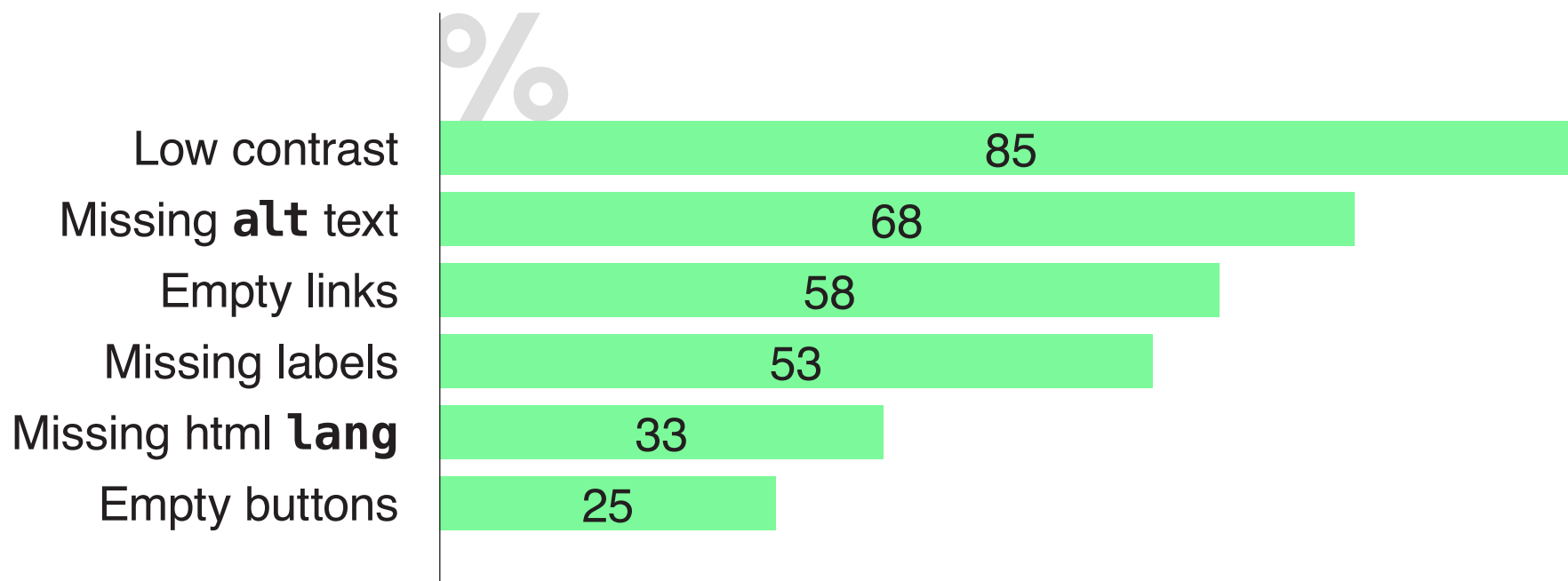


\* Impairment types by percentage declared by disabled people in FRS 2017/18, UK



**97.8% of home pages had detectable WCAG 2 failures.**

## Most common WCAG failures



\* Based on WebAIM automated accessibility evaluation of “top” million web sites conducted in February 2019

**Design for the needs  
of people with  
permanent, temporary,  
or situational  
disabilities.**

**Help users focus on  
core tasks and  
information.**

**Put  
people  
first.**

# The Basics.

Most common mistakes & how to fix them.

# 01

# Accessibility evaluation tools



Lighthouse

- available in **Chrome Dev Tools, Audits** tab
- a11y audits and suggestions for manual testing



VoiceOver

- Mac's built in **screen reader**
- activated with **Cmd + F5**



WAVE

- WebAIM's a11y evaluation tool
- available as a **browser extension**<sup>1</sup>

<sup>1</sup> <https://wave.webaim.org/extension>

# Non-semantic HTML5

```
// Some well defined React's JSX  
// But is it accessible?
```

```
<App>  
  <Header>  
    <Logo />  
  </Header>  
  <Content>  
    <Headline>Hello!</Headline>  
    <Text>👉</Text>  
  </Content>  
  <Footer />  
</App>
```





```
<div>
  <div>
    
  </div>
  <div>
    <div>Hello!</div>
    <div>👋</div>
  </div>
  <div>
    Legal stuff
  </div>
</div>
```

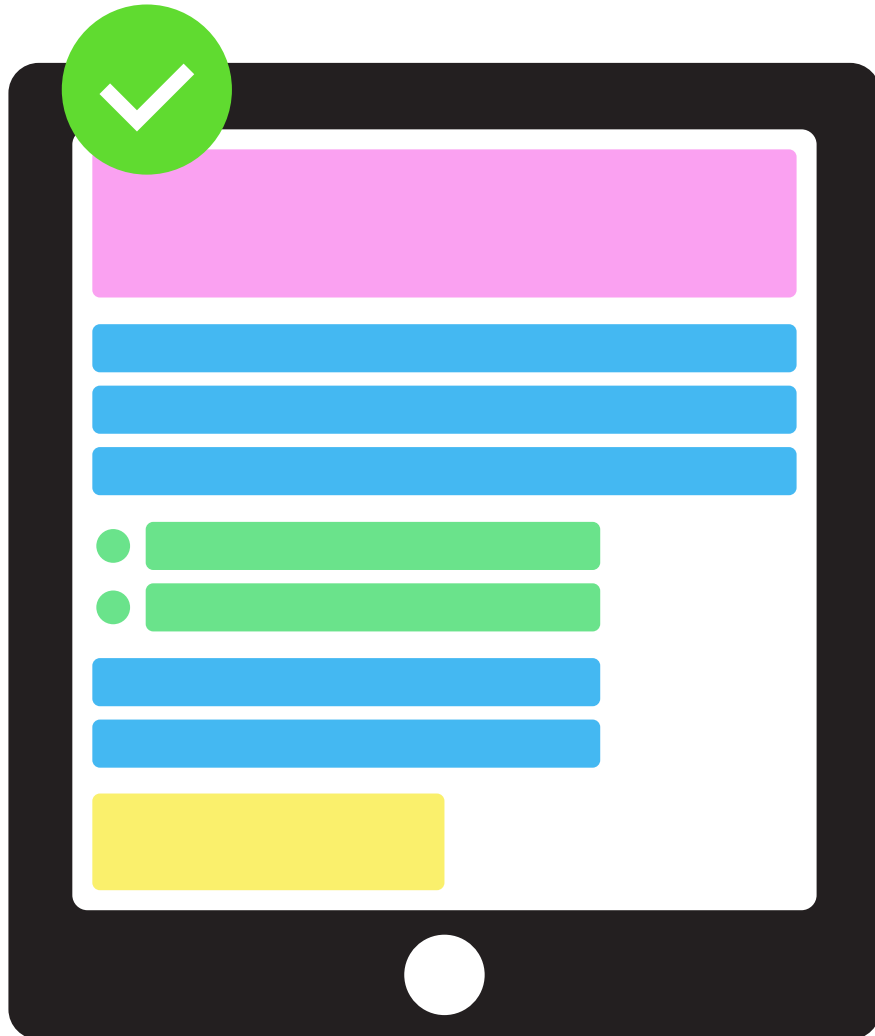


```
<div>
  <header>
    
  </header>
  <main>
    <h1>Hello!</h1>
    <p>👋</p>
  </main>
  <footer>
    Legal stuff
  </footer>
</div>
```

# Convolved content

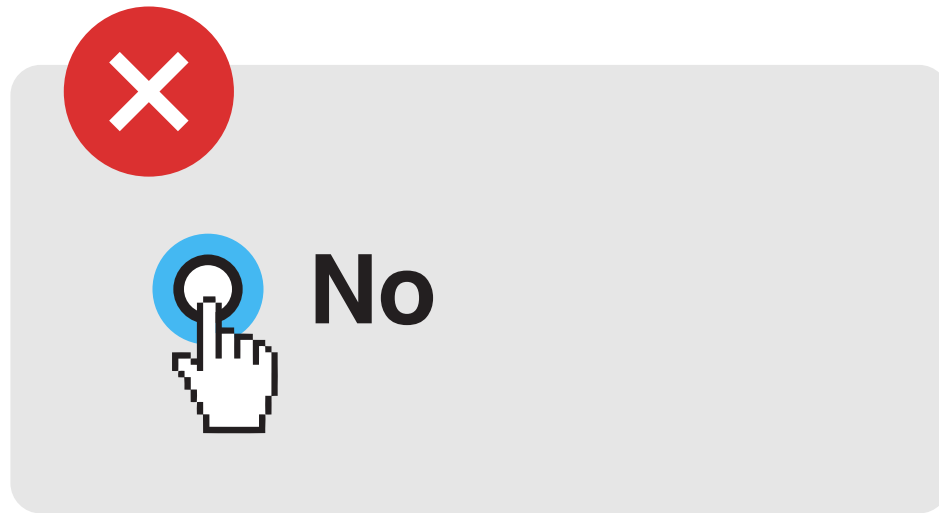


- make users scan long blocks of content
- use many figures of speech and idioms
- write walls of text
- build complex and cluttered layouts

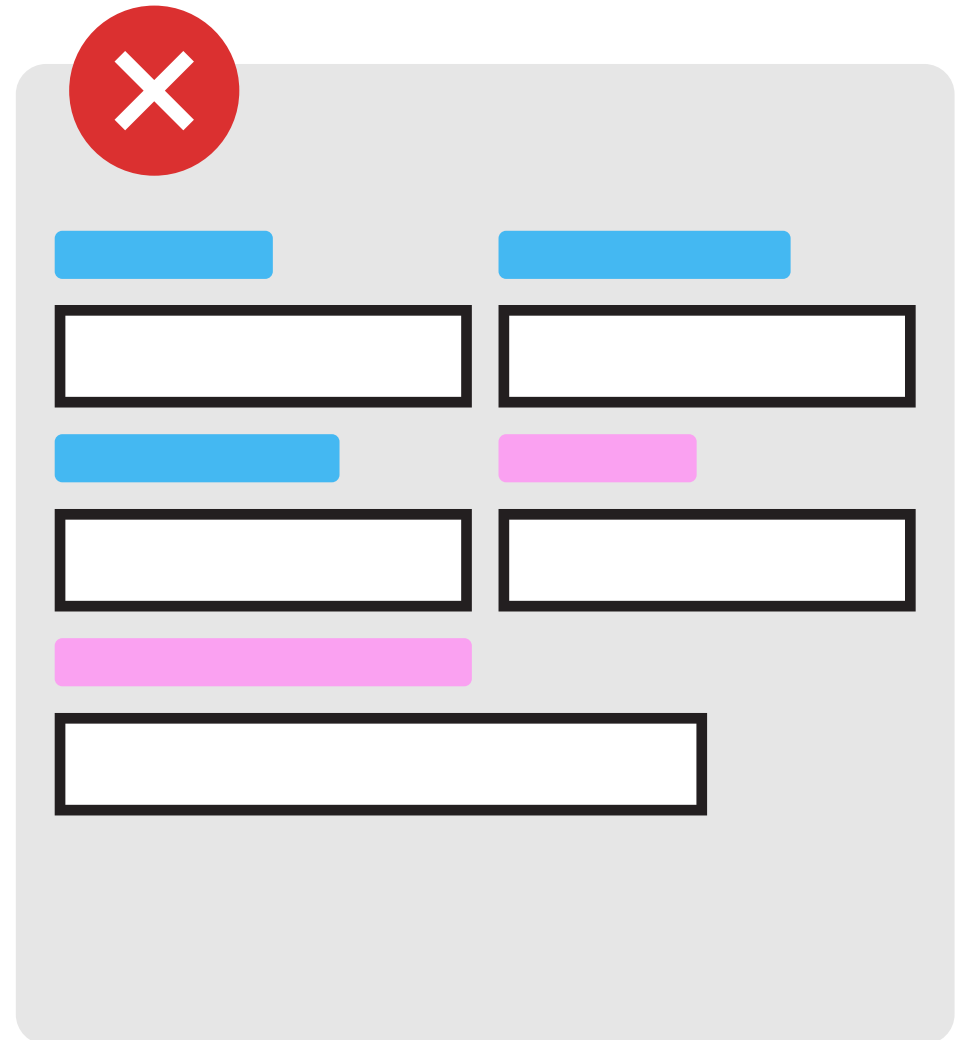


- keep content **short** and **clear**
- write in **plain English**
- use **bullets** instead of a wall of text
- build **simple** and **consistent** layouts

# Demand precision

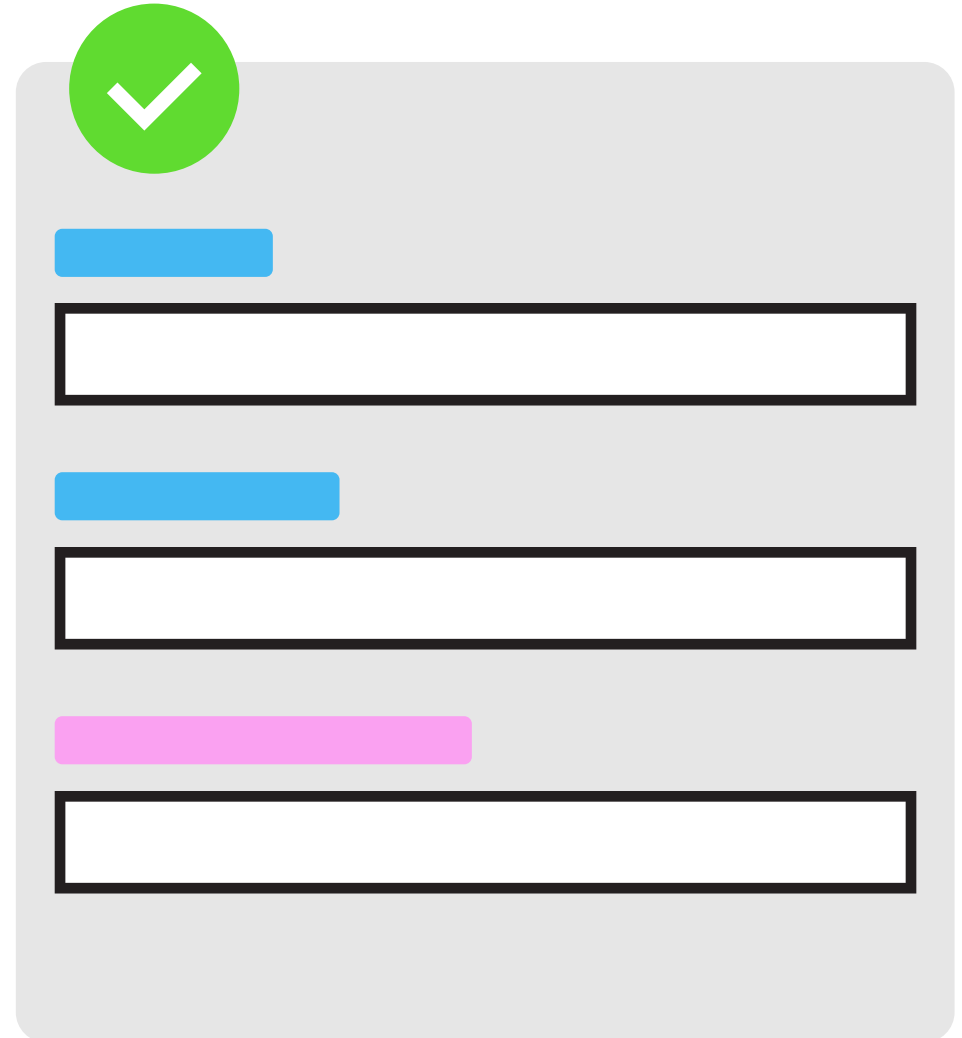


- bunch various interactions closely together
- require precise touch / cursor movements
- make complicated custom controls





- give your designs **breathing room**
- make clickable actions **large**
- use **native web elements**



```
import styled from "styled-components"
```

```
// Create basic reusable hitbox
```

```
const Hitbox = styled.button`
```

```
  min-width: 6rem;
```

```
  min-height: 6rem;
```

```
  cursor: pointer;
```

```
  border: none;
```

```
  background: none;
```

```
`
```

```
// And use it like that
```

```
const CustomActionButton = styled(Hitbox)`...`
```

# Low contrast



The Message



The Message

- design with high contrast in mind, validating color contrast with tools such as **Contrast Grid**<sup>1</sup>
- keep a color palette in **design tokens** to avoid mismatches in future
- size of the text matters

<sup>1</sup> <https://contrast-grid.eightshapes.com>

# Missing alt text



```

```



```

```



# Empty links



- never bake text into image
- if can't provide **alt** text for image within a link




```
<a href="/stickers-promo">  
    
</a>
```



```
<a href="/stickers-promo">  
    
</a>
```

# Missing form input labels



Submit



```
<form>
  <input
    type="text"
    name="email"
  >
  <button type="submit">
    Submit
  </button>
</form>
```



Email

Submit



```
<form>
  <label>
    Email
    <input
      type="text"
      name="email"
    >
  </label>
  <button type="submit">
    Submit
  </button>
</form>
```

# Missing html lang



```
<!DOCTYPE html>
<html>
  <head>...</head>
  <body>...</body>
</html>
```



```
<!DOCTYPE html>
<html lang="en">
  <head>...</head>
  <body>...</body>
</html>
```

# Empty buttons



```
<button  
  class="search__btn"  
></button>
```



```
<button  
  class="search__btn"  
  aria-label="Search"  
></button>
```

# General tips

Keep in mind these general guidelines:

- use **appropriate HTML elements** as much as possible since they'll have a lot of a11y behavior built-in
- use ARIA only when actually need to – it is easy to make things worse if you don't know what you're doing with it
- prefer **rem** or **em** CSS measurement units over **px** – it works better with magnification
- don't disable or override basic browser's functionality – complex custom controls are likely to **inconvenience someone**

# Amazing a11y

Resources worth checking:

- **WebAIM** website<sup>1</sup>, especially **Web Accessibility for Designers**<sup>2</sup> and **WCAG 2 Checklist**<sup>3</sup>
- **Inclusive Design Principles**<sup>4</sup> website
- **Accessibility Posters**<sup>5</sup> by GOV.UK

1 <https://webaim.org>

2 <https://webaim.org/resources/designers>

3 <https://webaim.org/standards/wcag/checklist>

4 <https://inclusivedesignprinciples.org>

5 <https://accessibility.blog.gov.uk/2016/09/02/dos-and-donts-on-designing-for-accessibility>



# Deep Dive.

Advanced a11y techniques.

# 02

# VoiceOver primer

Most useful shortcuts for website auditing:

**Cmd + F5**

turn it on / off

**Ctrl + Opt + Cmd + H**

navigate between headings<sup>1</sup>

**Ctrl + Opt + ← / →**

move across page elements

**Ctrl + Opt + Shift + ↓**

start interacting with a content of a page

**Ctrl + Opt + Space**

simulate click on an element

**Ctrl + Opt + U**

*rotor* menu (full list of links, headings, etc.)

**Ctrl**

stop talking

<sup>1</sup> most often used combination by real VO users after just landing on the page

# Proper structure of content

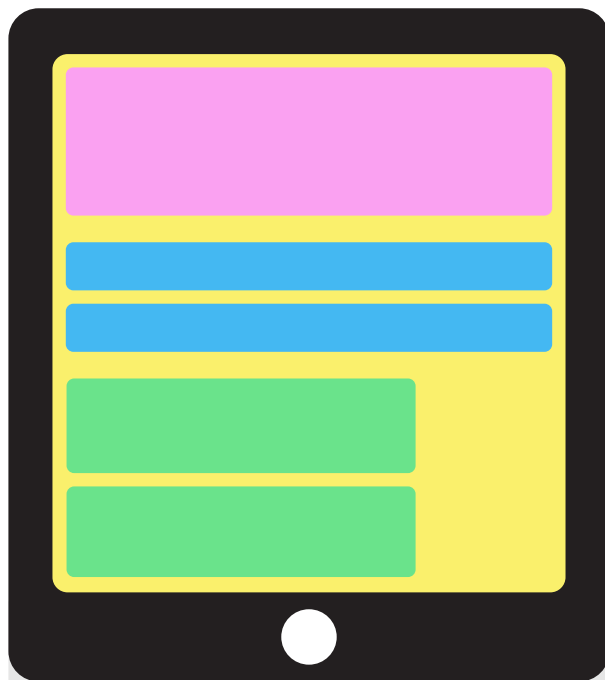
Say **NO** to wireframes. Plain text is enough. Focus on content details first.

A1 Landing Page							
	A	B	C	D	E	F	G
1	Landing Page						
2							
3	ORGANISM	LABEL	DESCRIPTION	CORE CONTENT	FEATURES / FUNCTION		
4	Header			App logo, Tagline	Link: Landing page		
5	Introduction	Daily Scrum is coming!	App description (max 10 words) with a SugarCat image	Headline, Subtitle, Image			
6	RoleButtons	Pick your role!	Set of buttons with backend, frontend and devops roles	Headline, Buttons / Links	Link: Backend dev page Link: Frontend dev page Link: Devops dev page		
7	Footer		Short text encouraging to contribute and visit careers page	Text, Crafted With Love link	Link: Careers page Link: Contribute Link: Crafted with love		
8							
9							
10							
11							
12							

\* <https://bigmedium.com/ideas/only-one-deliverable-matters.html>



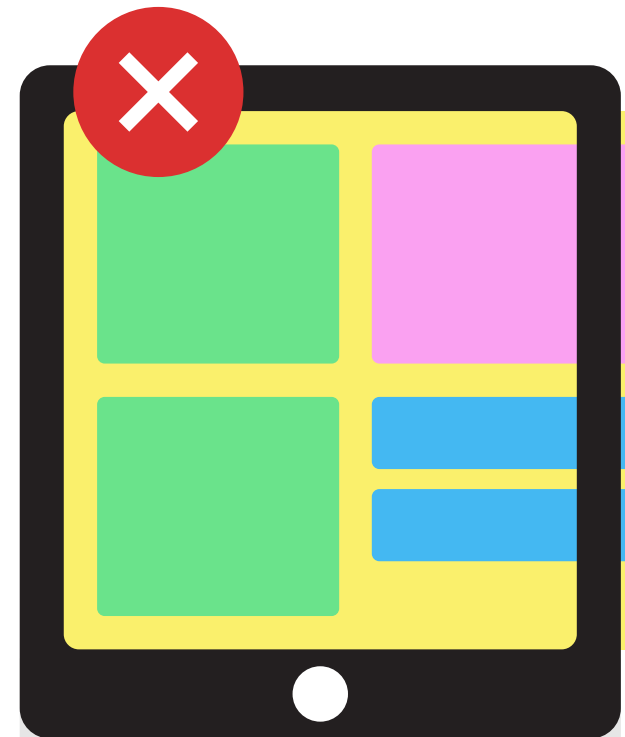
# Logical, linear layout



100% magnification

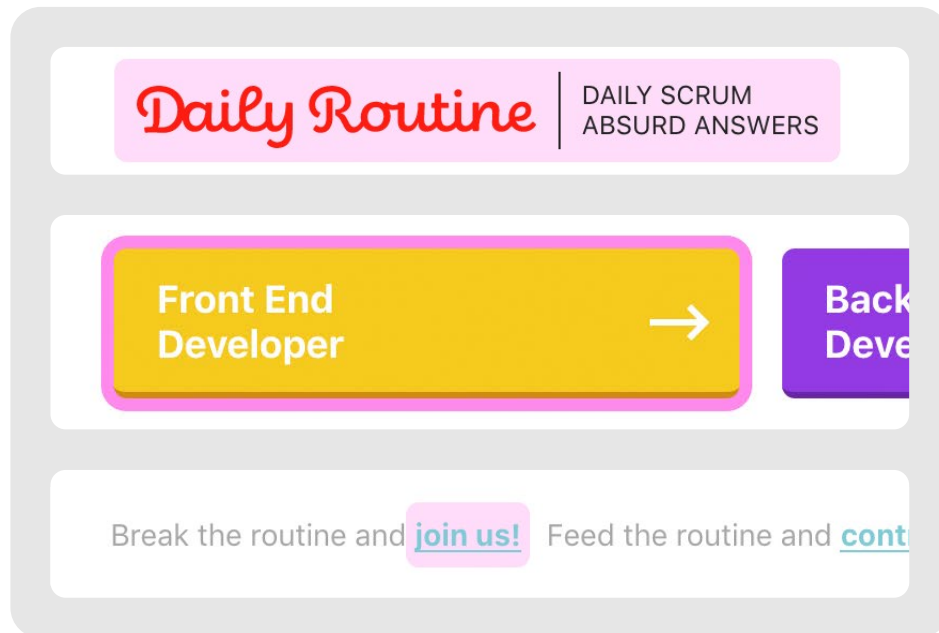


150% magnification



150% magnification

# Importance of *focus*



- to *focus* next element press a **Tab**
- to *focus* previous element use **Shift + Tab**
- focus state is **crucial** for keyboard navigation
- keep the focus style consistent
- remember about correct order of CSS selectors:  
**Link–Visited–Hover–Focus–Active**



```
button:focus {  
  outline: 0;  
}
```

- **don't remove default outline** if you are not planning to introduce custom style
- it is possible to style **outline** and it's valid option for **rectangular elements**





- alternatively use simple **border** if it is not used by other states like *hover*
- **outline** is still preferred since it changes the silhouette of an element



```
button:focus {  
  outline:  
    4px solid #ff8aed;  
  outline-offset: 0;  
}
```





- use **box-shadow** to simulate outline for elements with rounded corners
- it's heavy to compute so use it sparingly



```
button:focus {  
  outline: 0;  
  box-shadow:  
    0 0 0 4px #ff8aed;  
}
```

# A little issue with :focus



- in most cases show focus styles only **when navigating with keyboard**<sup>1</sup>
- **:focus-visible** comes to the rescue, unfortunately it's experimental feature hidden behind flag in most browsers<sup>2</sup>
- nice polyfill is available to install with **npm i -S focus-visible**
- sadly there is no similar alternative for **:focus-within**

<sup>1</sup> <https://github.com/WICG/focus-visible/blob/HEAD/explainer.md>

<sup>2</sup> [https://developer.mozilla.org/en-US/docs/Web/CSS/:focus-visible#Browser\\_compatibility](https://developer.mozilla.org/en-US/docs/Web/CSS/:focus-visible#Browser_compatibility)

```
// Import polyfill in the root of an app (index.js)
import "focus-visible/dist/focus-visible"

// Just use .focus-visible wherever you would
// normally use :focus, for example:
import styled from "styled-components"

const Hitbox = styled.button`
  ...

  &.focus-visible {
    outline: 4px solid #ff8aed;
  }
`
```

```
import styled from "styled-components"

// :focus-within could be used to give a meaningful
// focus styles to group of inputs

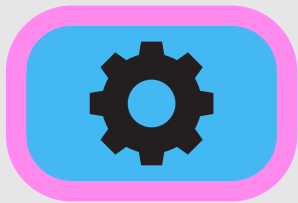
// Unfortunately there is no equivalent of :focus-visible
const RadioButtonsWrapper = styled.div`
  ...

  &:focus-within {
    outline: 4px solid #ff8aed;
  }
`
```

# Guiding screenreaders with ARIA



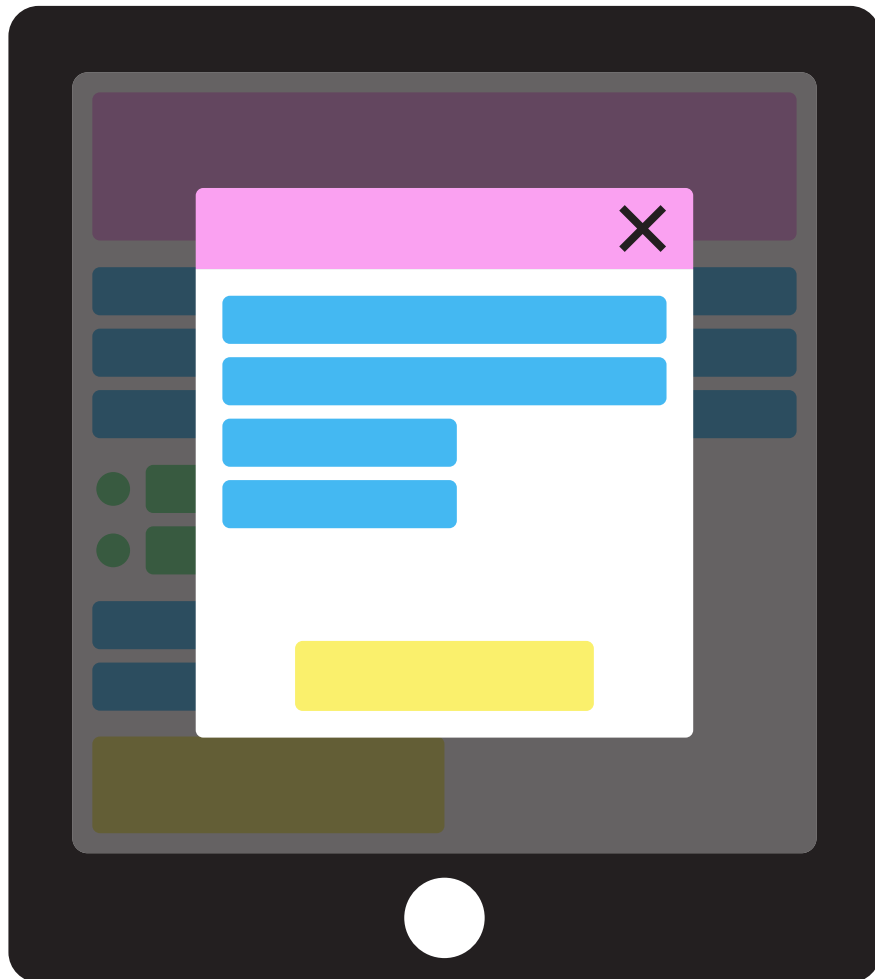
```
function SugarCatArtwork() {  
  return (  
    <svg  
      ...  
      role="img"  
      aria-labelledby="sugar-cat-char"  
    >  
      <title id="sugar-cat-char">  
        Sugar cat  
      </title>  
      ...  
    </svg>  
  )  
}
```



```
import Hitbox from "../Hitbox"

function SettingsButton({ onClick }) {
  return (
    <Hitbox
      aria-label="Open settings"
      onClick={onClick}
    >
      <SettingsIcon/>
    </Hitbox>
  )
}
```

# It's a trap



- in most situations users don't want to tab-out of modal
- sometimes it is required to *lock* user intention and focus
- no need to implement it on your own, just install handy library  
**`npm i -S react-focus-lock`**

```
import FocusLock from "react-focus-lock"

function Modal({ onClose }) {
  return (
    <FocusLock autoFocus={false}>
      <Header>
        <Headline>Settings</Headline>
        <CloseButton onClick={onClose} />
      </Header>
      ...
    </FocusLock>
  )
}
```



# Awesome a11y libraries

Some handy libraries worth checking:

- **focus-visible** – **:focus-visible** polyfill<sup>1</sup>
- **react-focus-lock** – useful for modals or other focused tasks<sup>2</sup>
- **react-focus-on** – locks focus, disables page scroll and hides rest of the page from screen readers<sup>3</sup>
- **lighthousebot** – make Lighthouse audits part of your CI/CD pipeline<sup>4</sup>

1 <https://github.com/WICG/focus-visible>

2 <https://github.com/theKashey/react-focus-lock>

3 <https://github.com/theKashey/react-focus-on>

4 <https://github.com/GoogleChromeLabs/lighthousebot>

**Thanks**  
**Dziękuję**  
**Grazie**  
**Obrigado**  
**Danke**  
**תודה**  
**谢谢**