



The illustration features four stylized characters in a vibrant, abstract space filled with geometric shapes like circles, squares, and triangles in shades of blue, orange, and teal. In the top center, a person with a ponytail lies on their back, using a laptop. To the left, a person in an orange jacket reaches up towards a floating digital interface displaying a list with checkboxes. On the right, a person in a teal jacket stands and holds a pen, with a squiggly line floating above them. In the bottom right, a person sits on a large teal sphere, interacting with a calendar grid that shows dates like 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, and 31. The central text 'Let's talk accessibility fundamentals.' is prominently displayed in a bold, black, sans-serif font.



**Let's talk accessibility  
fundamentals.**

## Accessibility fundamentals


# By the end of this talk



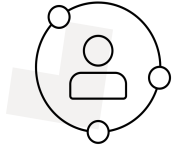
- + You will not be an accessibility expert. Sorry.
- + You should understand how WCAG and WAI-ARIA relate to accessibility.
- + You should have an idea of some areas where you can improve accessibility.
- + You will have one concrete way you can improve your code.



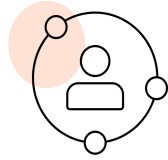
**Accessibility is the practice  
of making a product  
usable by as many people  
as possible.**



## Some people that benefit from accessibility.



**Color blindness**



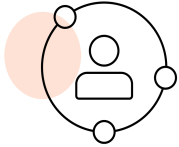
**Migraines**



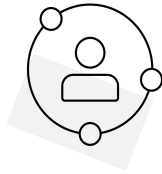
**Low-vision**



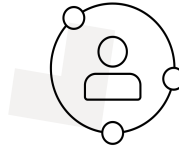
**Dark mode**



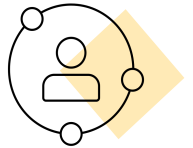
**Vertigo**



**Aging**



**Carpal Tunnel**



**Shortcut enthusiast**

# **The intersection of technical quality and user experience.**

## **Consistent & understandable**

An inconsistent design creates difficulties and frustration for all types of users. Accessibility encourages thoughtful decisions on grouping options, careful color choices, and simplified UI

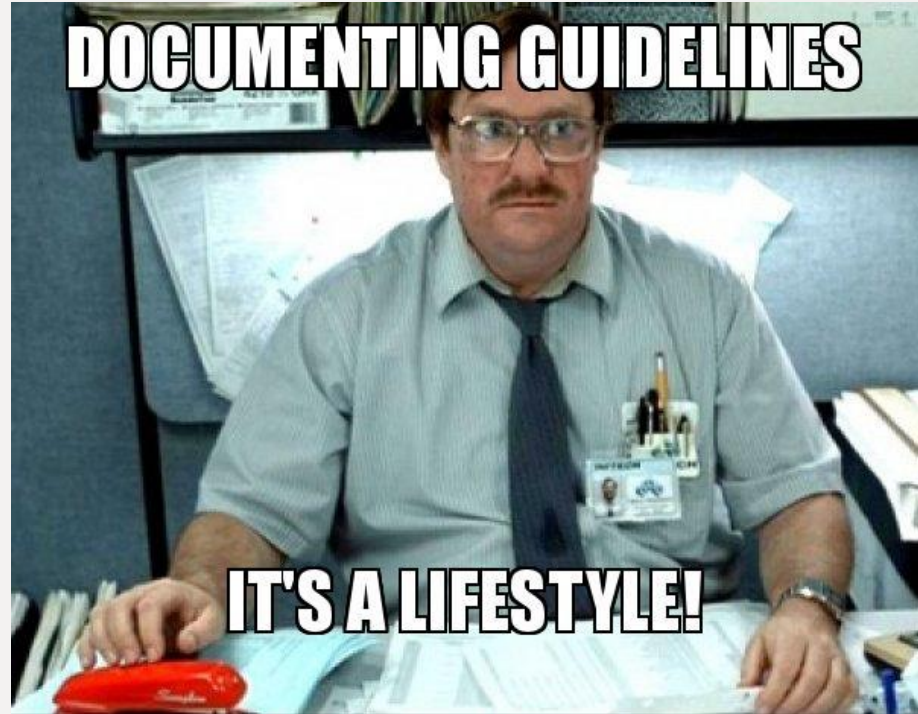
## **Programmatically identifiable**

WCAG standards requires well-formed code with additional identifiers for screen readers. This valid code makes it easier for testing frameworks and other robots to identify elements.

## **Easy to navigate**



Keyboard shortcuts are used by many users. ARIA recommendations provide clear guidelines for how each kind of component should behave.

Nice. But can I get a  
standard?






# **WCAG: Web Content Accessibility Guidelines**



**More than just an  
accessibility guideline, it  
provides testing protocol,  
techniques for success,  
and calls out frequent  
failures.**





WCAG 2.1

# The international standard



- + **Perceivable**, **operable**, **understandable**, and **robust** are the four guiding principles.
- + WCAG has three conformance levels but **Level AA** is what most laws will cite.
- + Includes [a very handy quick guide.](#)

**Great! Good to know.**



## WCAG Highlights

# Be precise



- [Validate your HTML](#)
- [Inputs have labels](#)
- [User is able to navigate to all interactive elements.](#)
- Keep the conversation going - communicate state changes.
- Use hex colors rather than opacity in CSS
- Display: hidden vs display: none

# Be thoughtful



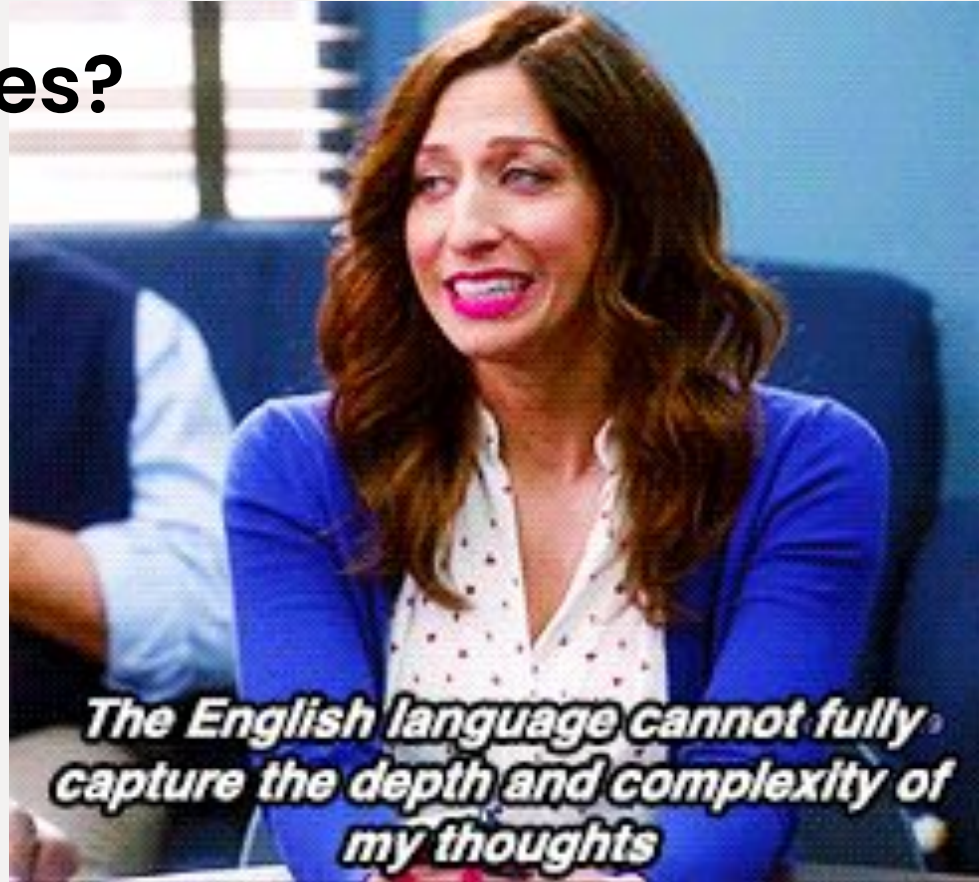
- Choose colors with appropriate contrast
- All focusable items have focus rings
- Use *prefers-reduced-motion* with animation
- Add alt attribute for images
- Ensure feature is responsive up to 400%

# Be honest



- Headings are for structure not styling
- Use labels that are specific and accurate.
- Use material honesty in designs
- Use native semantic elements when possible
- Ensure promises are kept.


What promises?



## VISUAL PROMISES

**Based on location, icons, hover behavior users expect certain behaviors.**

We expect buttons to trigger with enter. Details to show/hide. Selects to allow us to navigate between options with the up/down button. These expectations come from experience and how browsers interpret native HTML elements.



**Details**


+ Add to favorites

Choose a pet: Spider ▾

## SEMANTIC ELEMENTS

**Conveys the meaning, role, state, and the category of your content.**

Communicates this to the browser but more importantly it communicates to screen readers. So, our choices in markup should be considered 'user facing.'



HTML Demo: <details>

Reset

HTML CSS

```
1 details {
2   border: 1px solid #aaa;
3   border-radius: 4px;
4   padding: .5em .5em 0;
5 }
6
7 summary {
8   font-weight: bold;
9   margin: -.5em -.5em 0;
10  padding: .5em;
11 }
```

Output

► Details

**Note:** The common use of a triangle which rotates or twists around to represent opening or closing the widget is why these are sometimes called "twisties."



# <div>-ing it up.

We can't always use native semantic elements.

Enter WAI-ARIA.



# WAI-ARIA: Web Accessibility Initiative – Accessible Rich Internet Applications



WAI-ARIA

# Guidelines for adding semantic value to elements



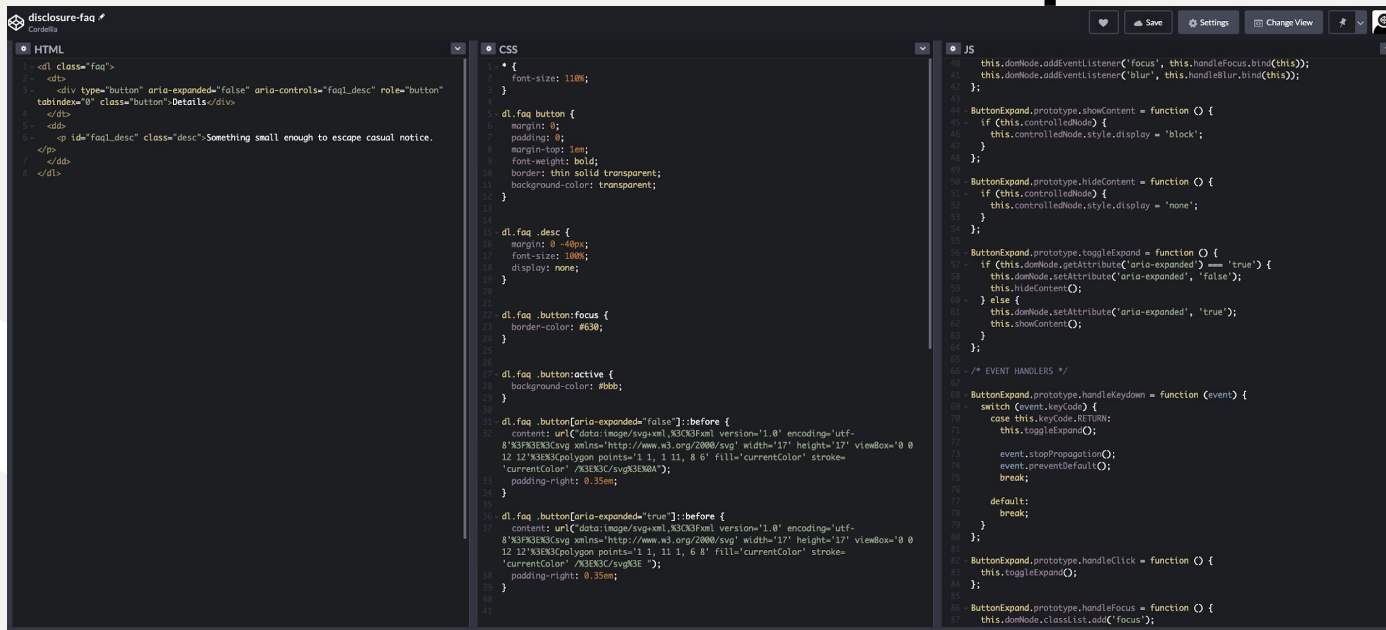
**Pattern example:** shows different way want to use this

**Keyboard support:** exactly what each interaction key should do.

**Role, Property, State, and Tabindex Attributes:** all listed out with usage notes.

[Semantic summary example in aria](#)

# Remember that <details> example?



```
disclosure-faq
HTML
<div class="faq">
  <div type="button" aria-expanded="false" aria-controls="faq_desc" role="button"
  tabindex="0" class="button">Details</div>
  <div id="faq_desc" class="desc">Something small enough to escape casual notice.
  </div>
</div>
CSS
/* {
  font-size: 110px;
}
dl.faq button {
  margin: 0;
  padding: 0;
  margin-top: 1em;
  font-weight: bold;
  border: thin solid transparent;
  background-color: transparent;
}
dl.faq_desc {
  margin: 0 -40px;
  font-size: 100%;
  display: none;
}
dl.faq_button:focus {
  border-color: #630;
}
dl.faq_button:active {
  background-color: #bbb;
}
dl.faq_button[aria-expanded="false"]::before {
  content: url("data:image/svg+xml,%3Csvg version='1.0' encoding='utf-8'%3E%3Csvg xmlns='http://www.w3.org/2000/svg' width='12' height='12' viewBox='0 0 12 12'%3E%3Cpolygon points='1 1, 1 11, 11 1, 11 11' fill='currentcolor' stroke='currentcolor'/%3E%3C/svg%3E%3C%3E");
  padding-right: 0.35em;
}
dl.faq_button[aria-expanded="true"]::before {
  content: url("data:image/svg+xml,%3Csvg version='1.0' encoding='utf-8'%3E%3Csvg xmlns='http://www.w3.org/2000/svg' width='12' height='12' viewBox='0 0 12 12'%3E%3Cpolygon points='1 1, 11 1, 11 11, 1 11' fill='currentcolor' stroke='currentcolor'/%3E%3C/svg%3E%3C%3E");
  padding-right: 0.35em;
}
JS
this.domNode.addEventListener('focus', this.handleClick.bind(this));
this.domNode.addEventListener('blur', this.handleClick.bind(this));
ButtonExpand.prototype.showContent = function () {
  if (this.controlledNode) {
    this.controlledNode.style.display = 'block';
  }
}
ButtonExpand.prototype.hideContent = function () {
  if (this.controlledNode) {
    this.controlledNode.style.display = 'none';
  }
}
ButtonExpand.prototype.toggleExpand = function () {
  if (this.domNode.getAttribute('aria-expanded') === 'true') {
    this.domNode.setAttribute('aria-expanded', 'false');
    this.hideContent();
  } else {
    this.domNode.setAttribute('aria-expanded', 'true');
    this.showContent();
  }
}
/* EVENT HANDLERS */
ButtonExpand.prototype.handleClickDown = function (event) {
  switch (event.keyCode) {
    case this.keyCode.RETURN:
      this.toggleExpand();
      event.stopPropagation();
      event.preventDefault();
      break;
    default:
      break;
  }
}
ButtonExpand.prototype.handleClick = function () {
  this.toggleExpand();
}
ButtonExpand.prototype.handleClickFocus = function () {
  this.domNode.classList.add('focus');
}
```

## Details

Something small enough to escape casual notice.

**Umm, that's a lot!**



WHOA!

# Most accessibility issues are caught in a couple steps




- 1 Ask for design accessibility review in [#qa-accessibility](#)
- 2 [Validate rendered HTML](#)
- 3 Address serious and critical errors from [axe chrome extension](#) scan.
- 4 Keyboard through the UI with a [critical eye](#).



**#1 CONCRETE TIP!**



## #1 TIP!



Don't put `<div>` inside `<span>`. [Know your content model restrictions!](#) `<span>` `<button>` can only have phrasal content.

If you declare a role on a div using ARIA the content model for the native element applies.





Don't put `<div>` content

#1

Error Element `div` not allowed as child of element `span` in this context. (Suppressing further errors from this subtree.)  
From line 1, column 16139, to line 1, column 16237  
`index="0"><div class="rounded-big col-12 card-hover-darken1 darken1-hover animate flex items-center pointer"><div c`

Contexts in which element `div` may be used:  
Where [flow content](#) is expected.  
As a child of a [dl](#) element.  
Content model for element [span](#):  
[Phrasing content](#).

only

... in a div using ARIA the  
the native element applies.

#1

in this context. (Suppressing further errors from this subtree.)

-hover animate flex items-center pointer"><div c

. (Suppressing further errors from this subtree.)

Error Element `div` not an  
From line 1, column 16139, to line 1, column

`index="0"><div class="rounded-big col-1`  
Contexts in which element `div` may be used:

Where [flow content](#) is expected.

As a child of a [div](#) element.

Content model for element [span](#):

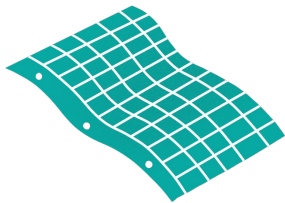
[Phrasing content](#).

... a div using ARIA the  
... the native element applies.



**What is next?**

# Airtable empowers you to build the tools you need based on the winning formula only you know



## Read

Learn everything you can  
about accessibility.



## Reference

Get familiar with  
WAI-ARIA patterns and  
Storybook contributions  
that have been reviewed



## Share

Join #a11y share any  
successes, challenges,  
and failures there!



**Questions?**