Accessibility Issues

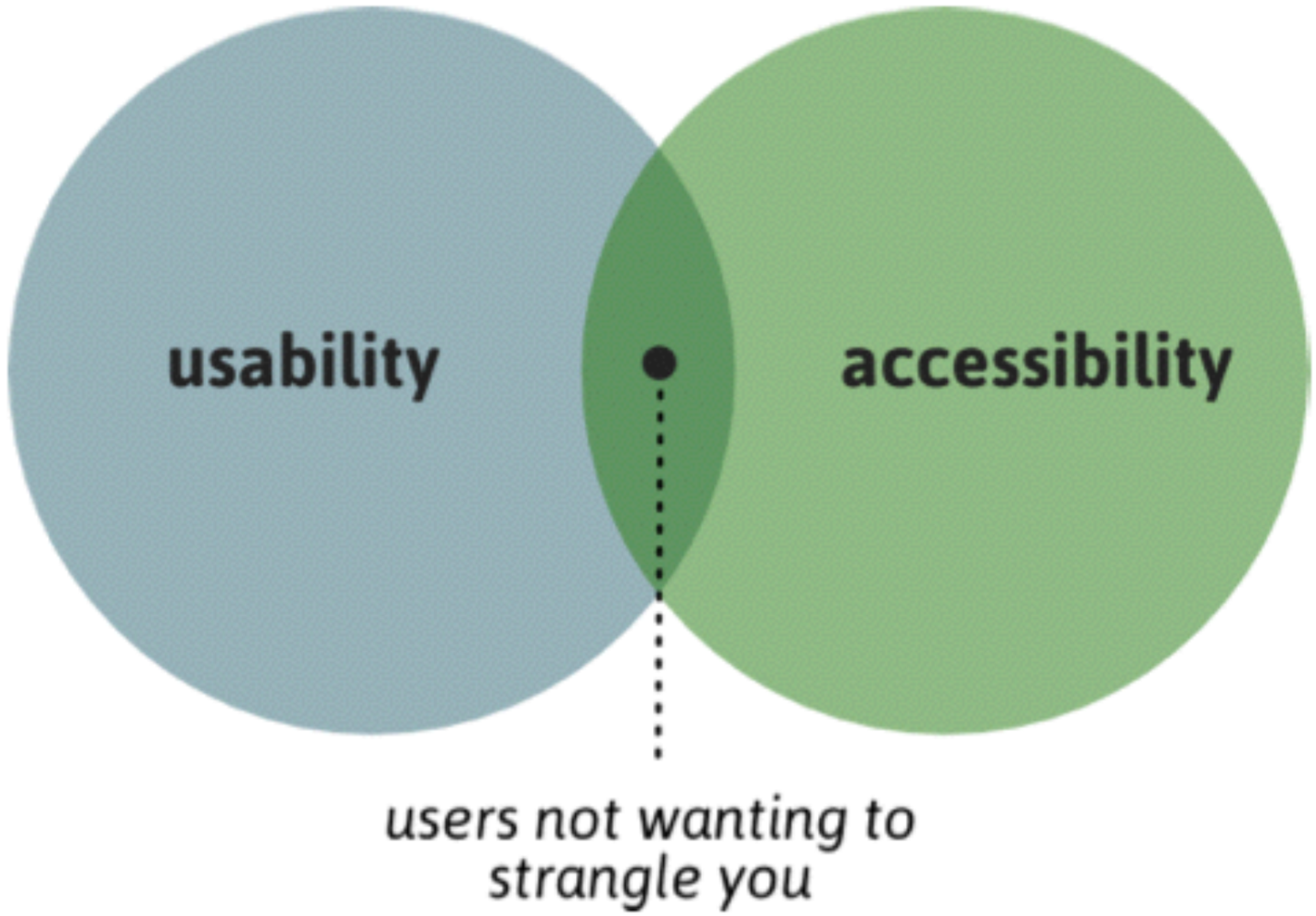
> Accessibility is not something we add to a website, but something we *start with and risk losing* with every enhancement. It is to be *retained*.

— *Scott Jehl*

https://twitter.com/scottjehl/status/411237303579721728

Accessibility: The Foundation

- Most guidelines are rather about *not* making things *inaccessible* than making things *accessible*.
  - *WAI-WCAG 2.09 (Web Content Accessibility Guidelines)*
    Set of general, basic accessibility recommendations.
  - *WAI-ARIA 1.0 (Accessible Rich Internet Applications)*
    Advanced, ambitious guide that takes a11y to a new level.
- *Goal:* keep the content usable by *keyboard users* and *screen readers users*, e.g. buttons and widgets.


usability
accessibility
users not wanting to
strangle you

Accessibility Techniques

- Accessibility is about defining *relationships,* properties and behaviors of elements explicitly.
  - *Buttons* have different states; they control elements,
  - *Forms* require concise and appropriate feedback,
  - *Widgets* have complex properties and behaviors,
  - *Navigation* requires appropriate sectioning.
- *Problem:* misconceptions about how users use screen readers/keyboards—and *browser support.*

Accessibility Techniques

- *Problem:* misconceptions about how users use screen readers/keyboards—and *browser support.*
  - Accessibility tools *do have JavaScript support*,
  - Screen readers can't *predict structure or purpose,*
  - Accessibility builds *on top of* existing experience,
  - *ARIA is well-supported*; native controls are better.

> *Semantic HTML* is markup that makes a positive contribution to the meaning of the page in a standardized way which can be understood by a *maximal range of user agents...*

— Heydon Pickering

"Apps for All"

> ...As a result, screen readers can understand the patterns and provide an equivalent experience of the same content both visually and aurally. Semantics is embedded in *attributes and HTML elements*.

— *Heydon Pickering*

"Apps for All"

Accessibility Techniques

- *Problem:* misconceptions about how users use screen readers/keyboards—and *browser support.*
  - Accessibility tools *do have JavaScript support*,
  - Screen readers can't *predict structure or purpose,*
  - Accessibility builds *on top of* existing experience,
  - *ARIA is well-supported*; native controls are better.

Accessibility Techniques

- Accessibility is about defining *relationships,* properties and behaviors of elements explicitly.
  - *Buttons* have different states; they control elements,
  - *Navigation* requires appropriate sectioning.
  - *Forms* require concise and appropriate feedback,
  - *Widgets* have complex properties and behaviors,
- *Problem:* misconceptions about how users use screen readers/keyboards—and *browser support.*

Accessible Buttons

- Only <button> should look like a button; classes on <a>, , aren't conducive to accessibility.
- HTML:

```
<button type="button">Read more…</button>
/* Beware: implicit type is 'submit', so be specific. */

<a
<span
```

- To activate a button, keyboard users must first be able to *focus* it. Screen readers rely on *aural feedback.*

Accessible Buttons

- To activate a button, keyboard users must first be able to *focus* it. Screen readers rely on *aural feedback.*
- <button> receives focus, can be operated via keyboard and is announced as “button” by screen readers.
- *Hint: perceived tactility* of interactive controls helps to communicate their utility. Big, bold, clickable.

Accessible Buttons

- <button> receives focus, can be operated via keyboard and is announced as “button” by screen readers.
- Buttons can have different *states:*
  - — *Pressed/unpressed* buttons,
  - — *Disabled* buttons,
  - — Buttons which *control other elements*,
  - — Buttons that *show/hide* parts of the page.
- *Hint: perceived tactility* of interactive controls helps to communicate their utility. Big, bold, clickable.

- HTML:

```html
<button type="button">Save</button>
```

- CSS:

```css
button {
    background-color: DarkSlateBlue;
    border-radius: 0.25em;
    box-shadow: 0 4px 0 #222; }

button:hover, button:focus {
    /* Make it look like you can press it */ }
```

- CSS:

```
button:active {
    position: relative;
    top: 3px; /* 3px drop */
    box-shadow: 0 0 0 #222; /* less by 3px (to zero) */
 }

button[disabled] {
    /* Styles for a disabled button */
    pointer-events: none;
    /* or cursor: not-allowed; */
}
```

- Nested text *within* <button> will be read out by screen readers when the button is accessed.
- For <input> buttons, the *value* attribute will be announced. Both should *never* be left empty.
- *WAI-ARIA* enhancements are always required when dealing with non-textual content.

- HTML:

```
<button aria-label="Undo">&#xE000;</button>

<p class="tooltip hidden">You can restore defaults
with <strong id="undo-text">Undo</strong> button.</p>
<button aria-labelledby="undo-text">&#xE000;</button>
```

ARIA Roles, Properties, States

- WAI-ARIA is a gap-filler for HTML semantics.
  It defines three kinds of accessibility attributes:

  — *Roles* define the meaning of an element
  (role="tooltip", role="tablist", role="search"),

  — *Properties* describe its characteristics
  (aria-required, aria-controls, aria-label, aria-labelledby),

  — *States* describe its current interaction states
  (aria-checked, aria-expanded, aria-hidden, aria-required).

- HTML:

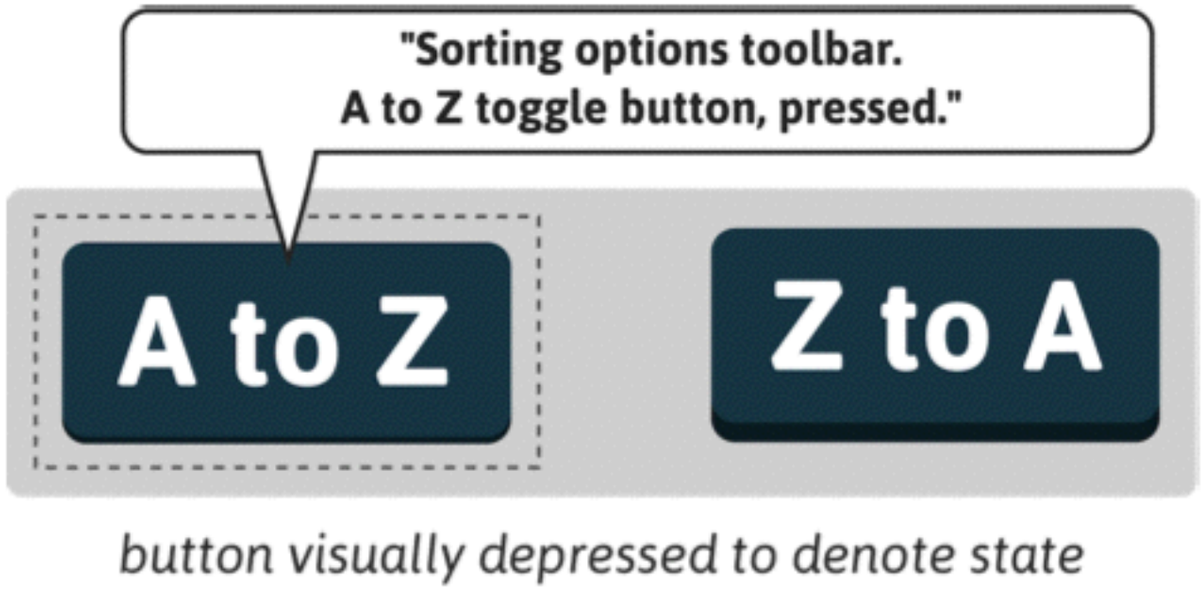
```
<button aria-label="Undo">&#xE000;</button>

<p class="tooltip hidden">You can restore defaults
with <strong id="undo-text">Undo</strong> button.</p>
<button aria-labelledby="undo-text">&#xE000;</button>
```

- HTML:

```
<div role="button" aria-label="Undo"><img
src="undo_icon.png" alt="Undo" /></div>

<p aria-hidden="true" class="tooltip hidden">Undo</p>
```

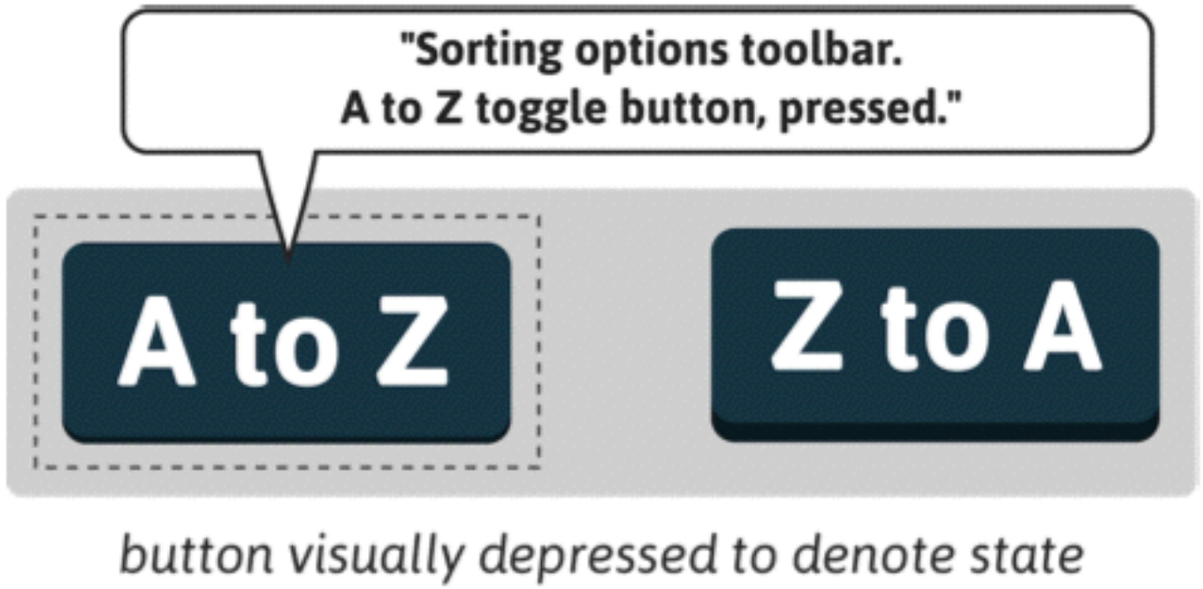
- Legacy code can be *augmented* with ARIA roles to patch code that is otherwise unsemantic.
- Using ARIA-roles should be a method of last resort, as it fully *overrides* native semantics.

button visually depressed to denote state



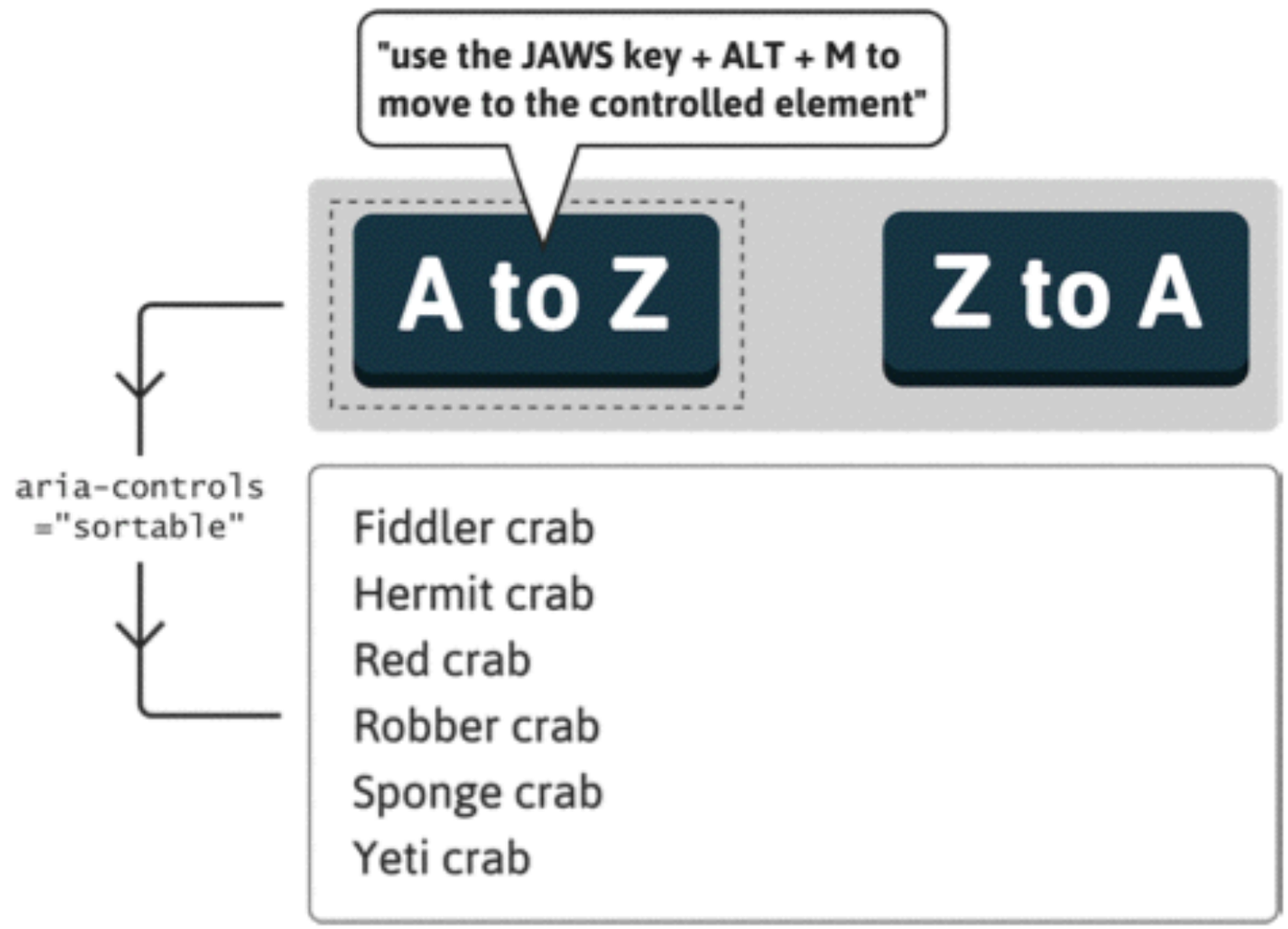
- HTML:

```html
<div role="toolbar"
     aria-label="sorting options"
     aria-controls="sortable">
        <button aria-pressed="true">A to Z</button>
        <button aria-pressed="false">Z to A</button>
</div>

<ul id="sortable" tabindex="-1">
    <li>Fiddler crab</li> <li>Hermit crab</li> ...
</ul>
```



- CSS:

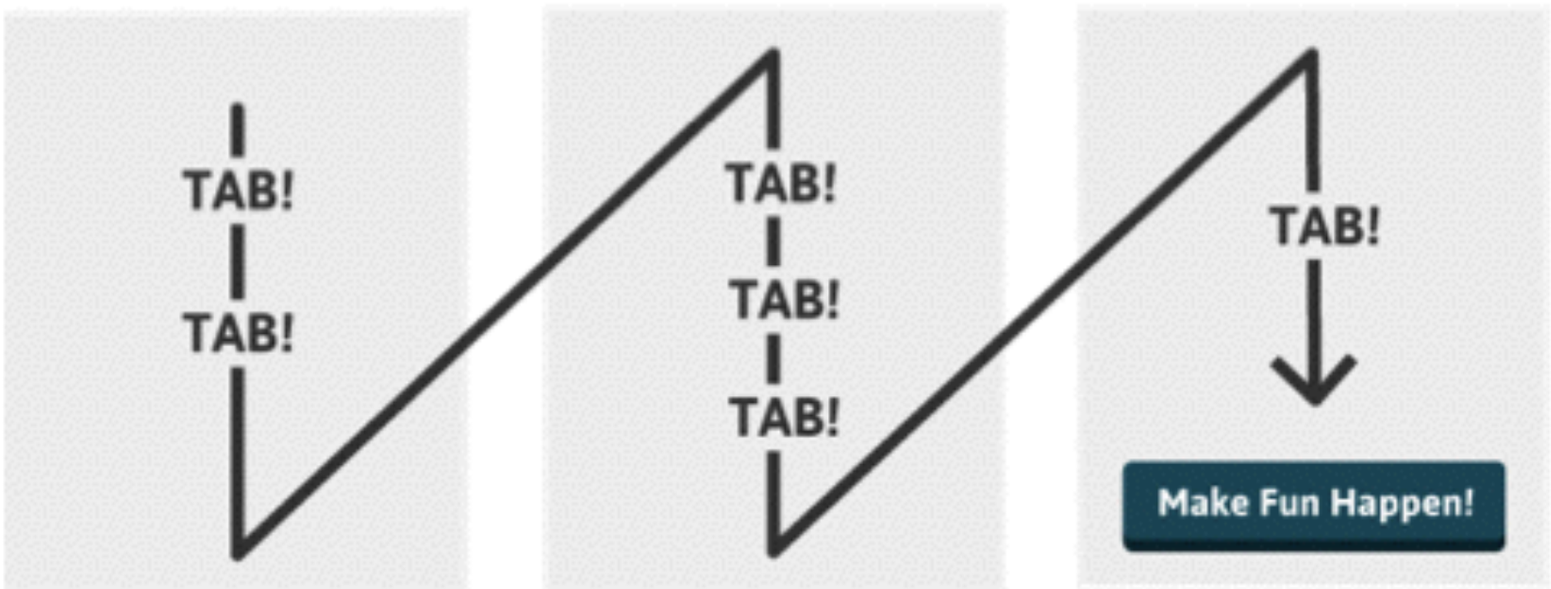
```
button:active,
button[aria-pressed="true"] {
    position: relative;
    top: 3px; /* 3px drop */
    box-shadow: 0 0 0 #222; /* less by 3px (to zero) */
}
```

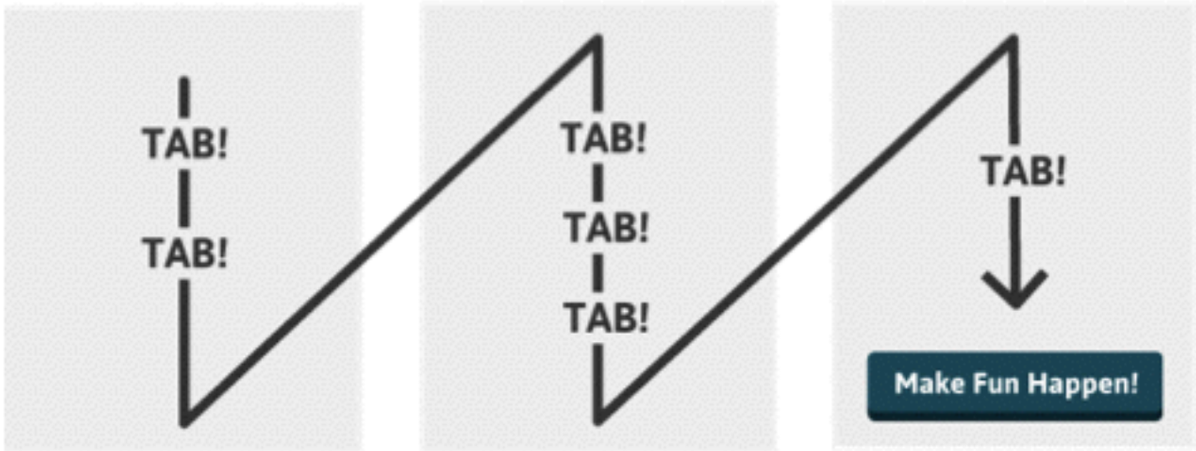

"use the JAWS key + ALT + M to move to the controlled element"
A to Z
Z to A
aria-controls ="sortable"
Fiddler crab
Hermit crab
Red crab
Robber crab
Sponge crab
Yeti crab

Accessibility Techniques

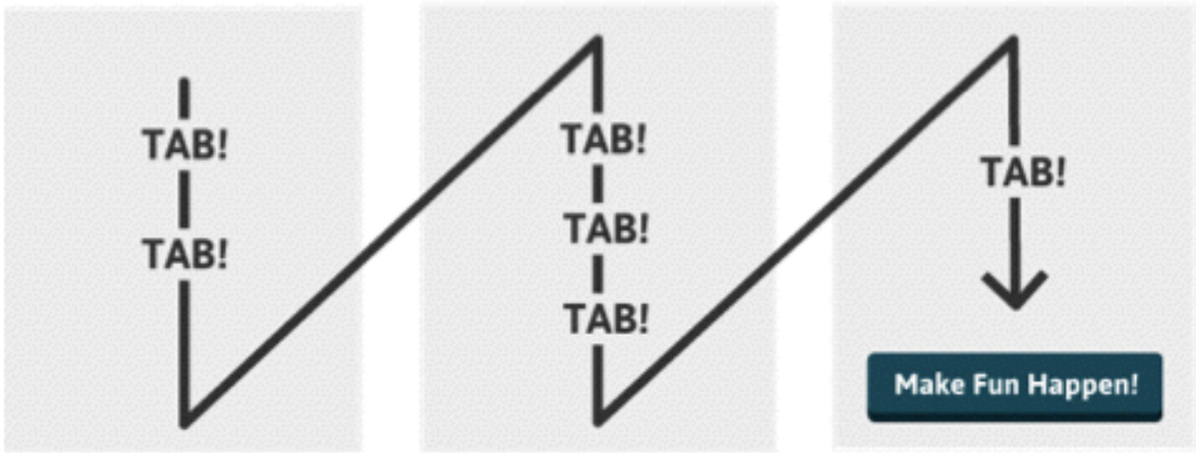
- Accessibility is about defining *relationships,* properties and behaviors of elements explicitly.
  - *Buttons* have different states; they control elements,
  - *Navigation*
  - *Forms*
  - *Widgets*
- *Problem:*

  screen readers/keyboards—and

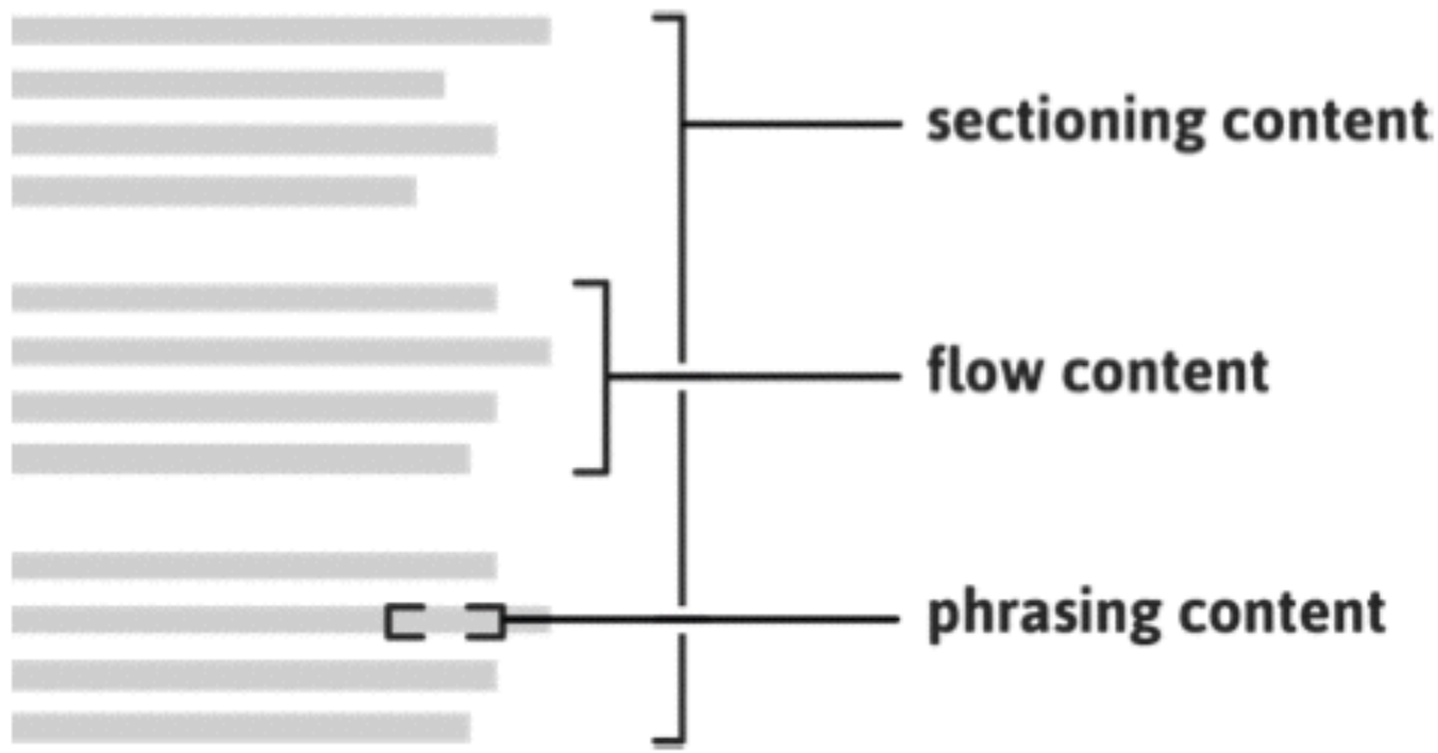

TAB!
TAB!
TAB!
TAB!
TAB!
TAB!
Make Fun Happen!

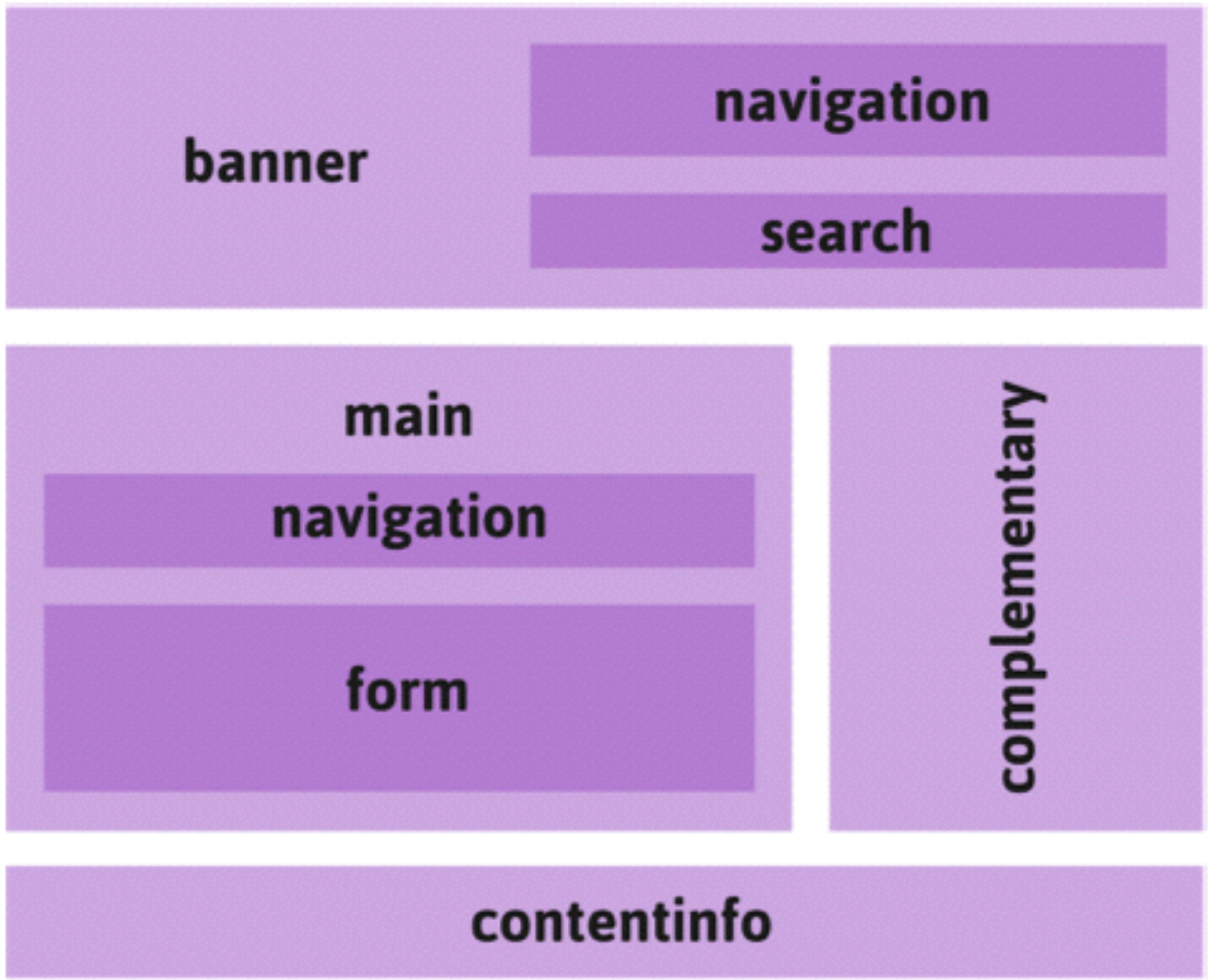


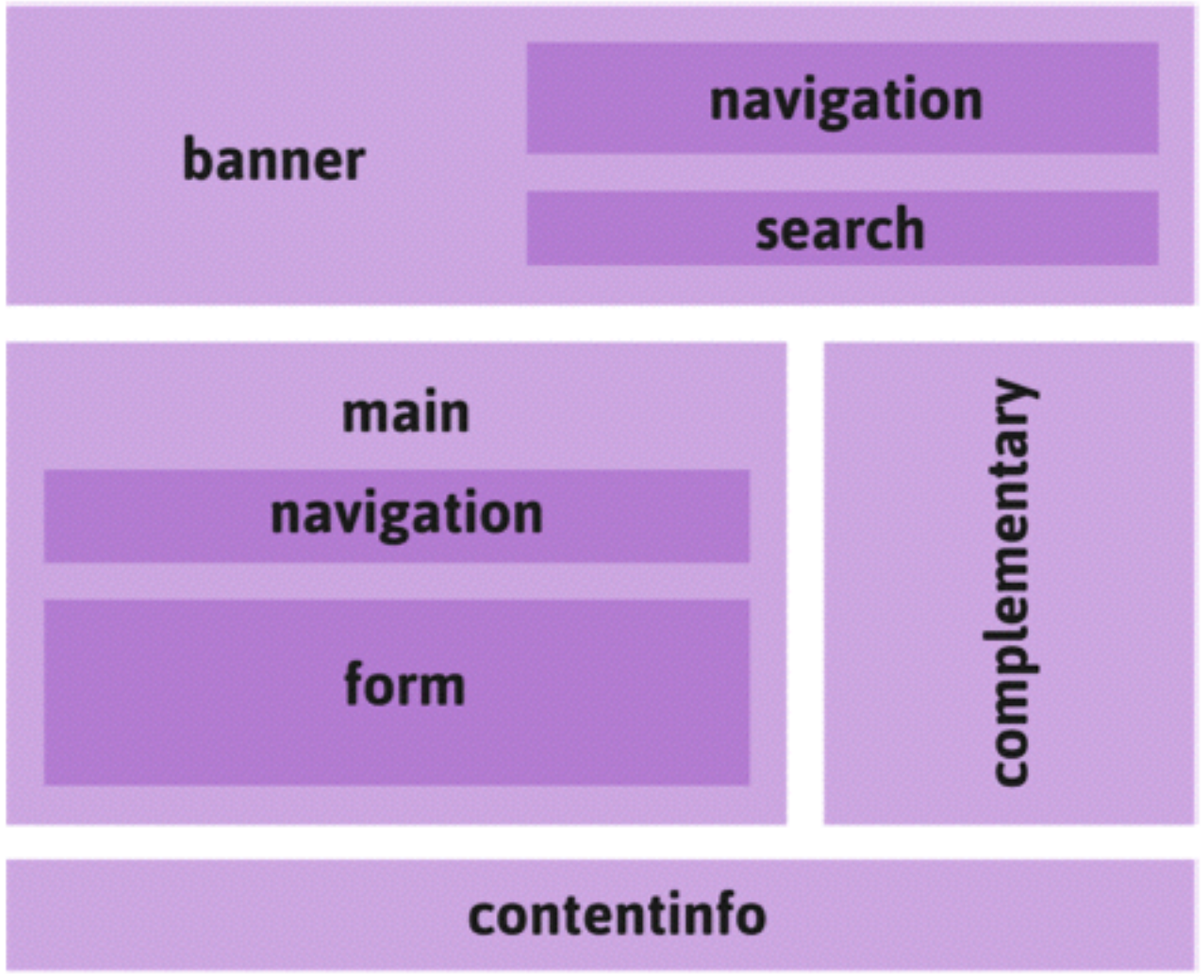
- *Links* can be focused by screen readers; they can distinguish between “Link” and “Same page link”.
- But *grid* is *invisible* to screen readers. Keyboard users don’t benefit from it either, they have to cover the same distance, *sideways*.



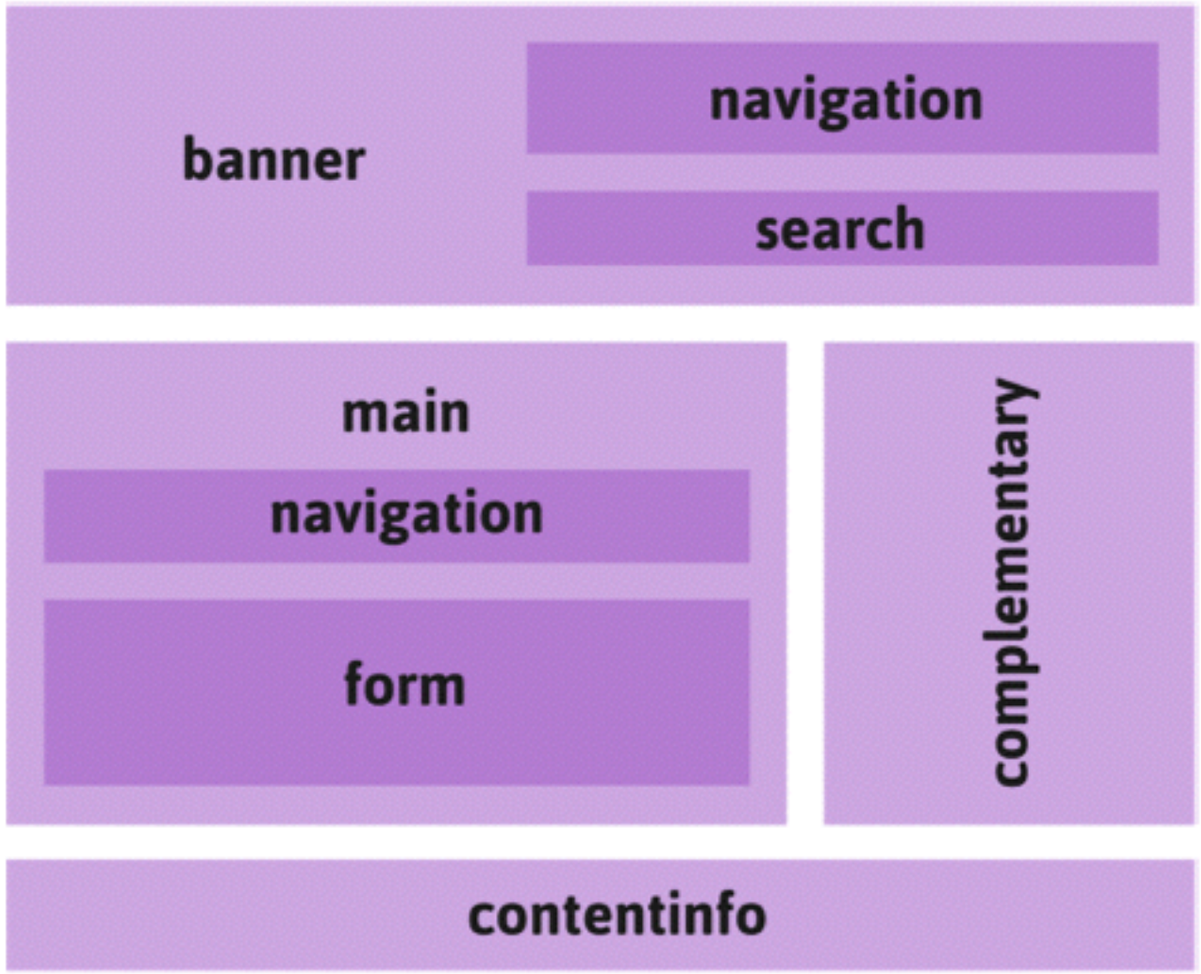
- But *grid* is *invisible* to screen readers. Keyboard users don't benefit from it either, they have to cover the same distance, *sideways.*
- *Solution:* draw an *invisible map* of the page by sectioning content and providing landmarks.


sectioning content
flow content
phrasing content


banner
navigation
search
main
navigation
form
complementary
contentinfo

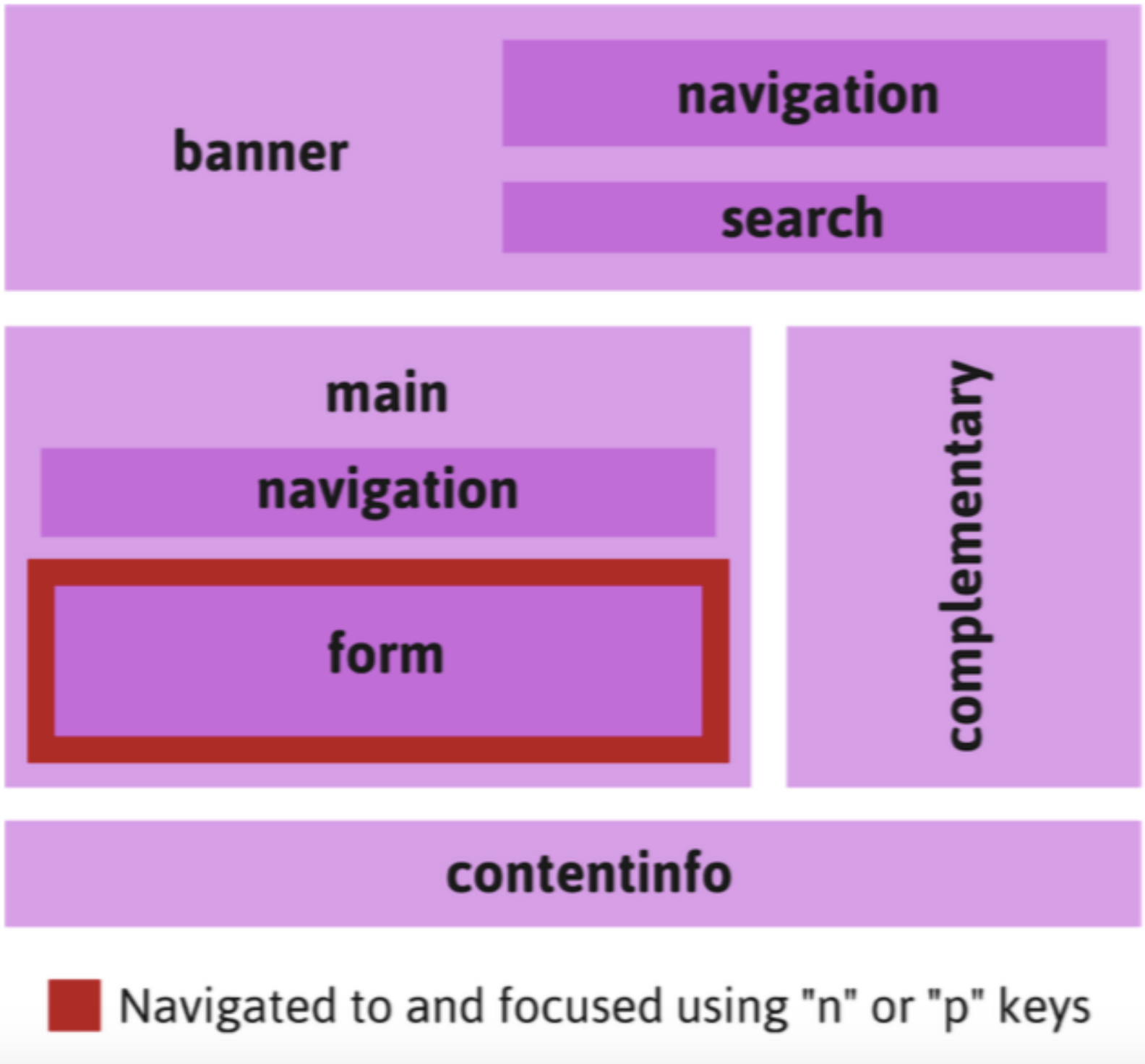


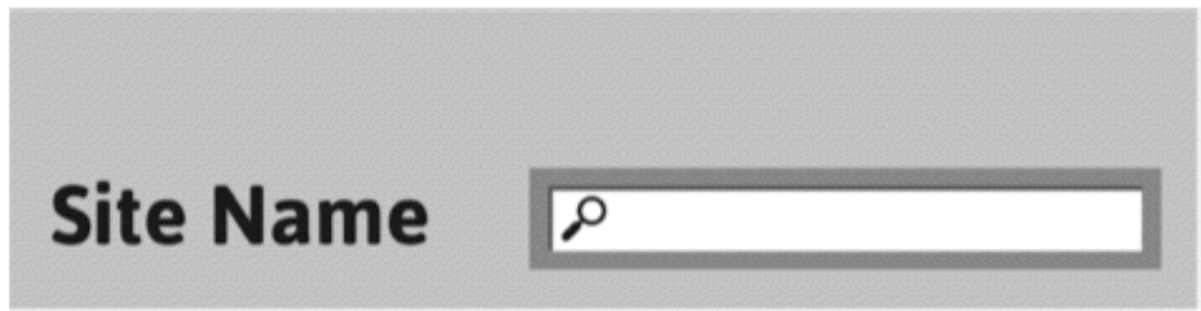
- *Sectioning* conveys structure to screen readers and keyboard users. But: avoid too much nesting!
- *Meaningful, consistent headings* and landmark roles help users jump between sections seamlessly.



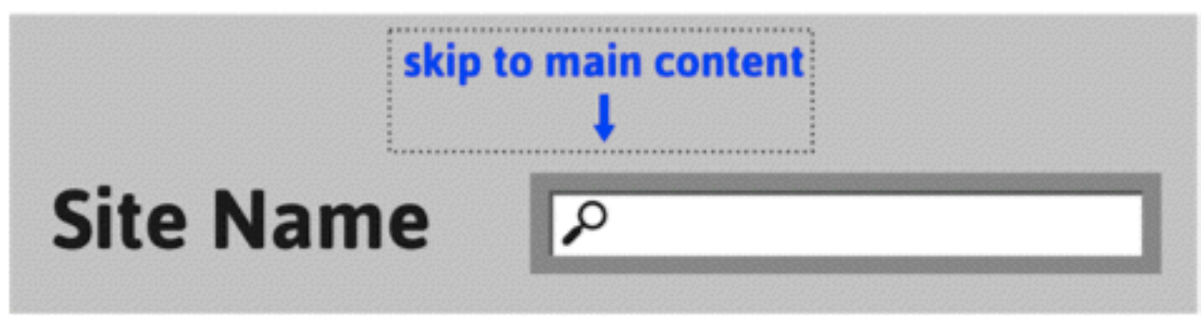
- *banner* (role="banner")
- *contentinfo* (role="contentinfo")
- *main* (role="main")
- *navigation* (role="navigation")
- *search* (role="search")
- *form* (role="form")
- *complementary* (role="complementary")

Applicable like *<div role="search"... >* to any HTML element.

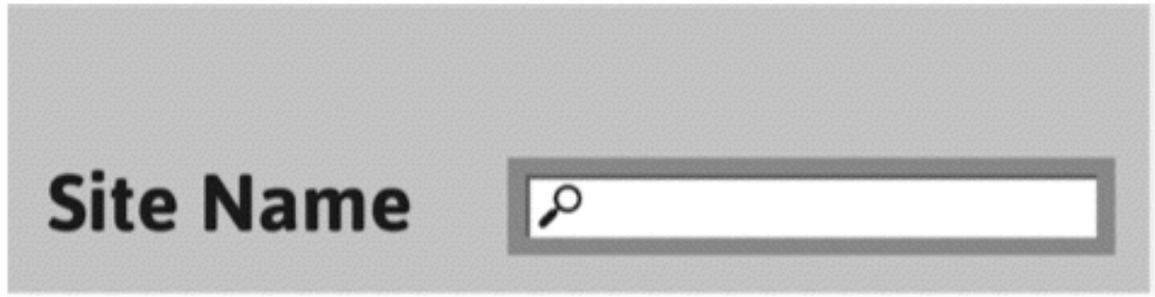

banner
navigation
search
main
navigation
form
complementary
contentinfo
Navigated to and focused using "n" or "p" keys



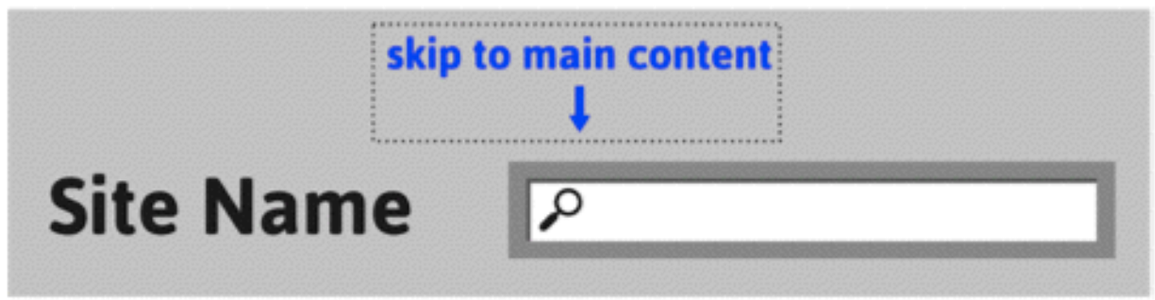
a.skip



a.skip:focus



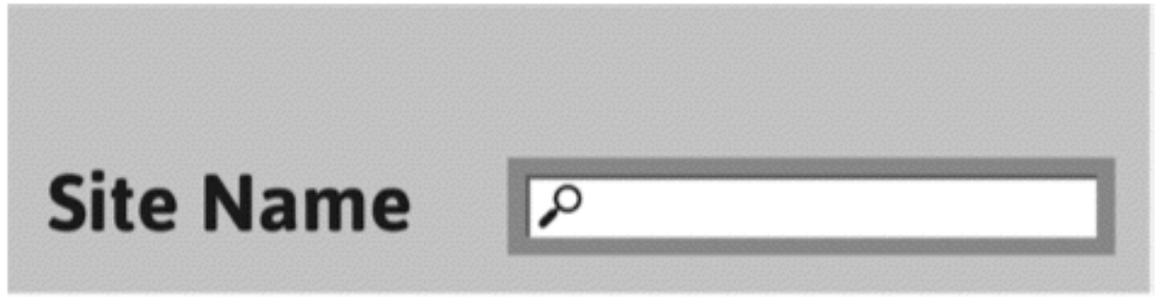
a.skip



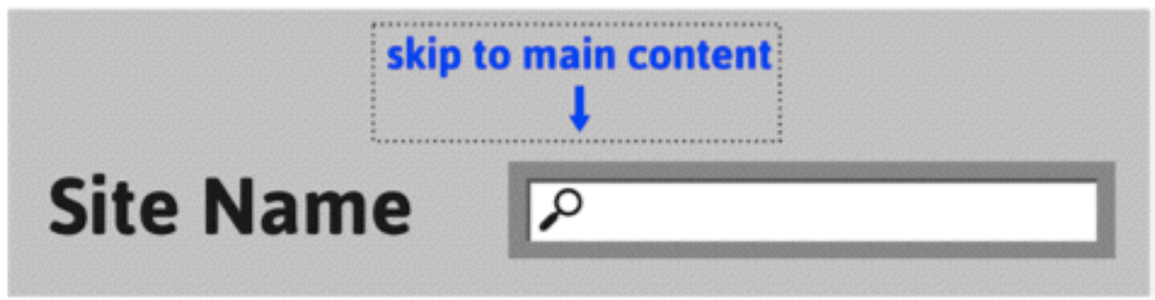
a.skip:focus

- *"Skip to content" links* often coded incorrectly:

```
a.skip {
    display: none; /* Nobody can see it. */
}
```



a.skip



a.skip:focus

- *"Skip to content" links* better pushed off-screen:

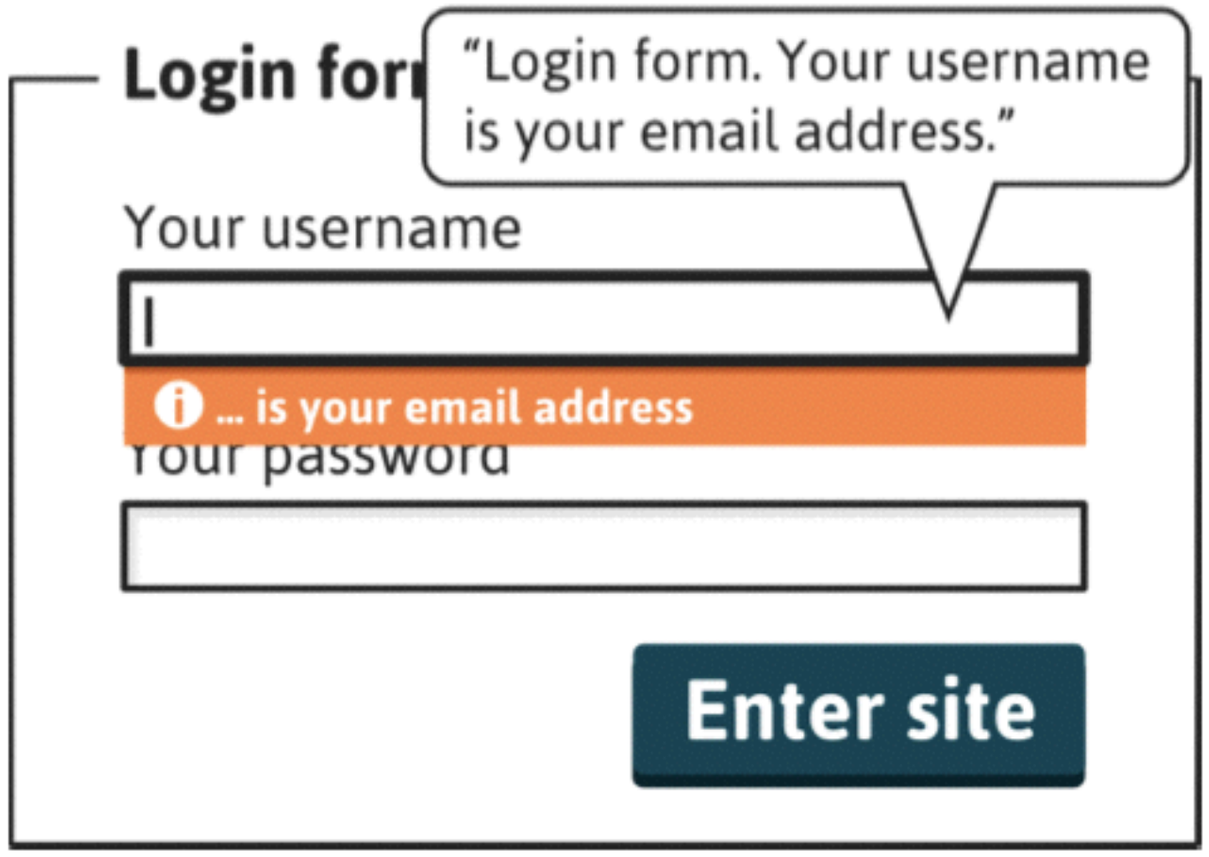
```
a.skip {
    position: relative;
    top: -100px;
}
```

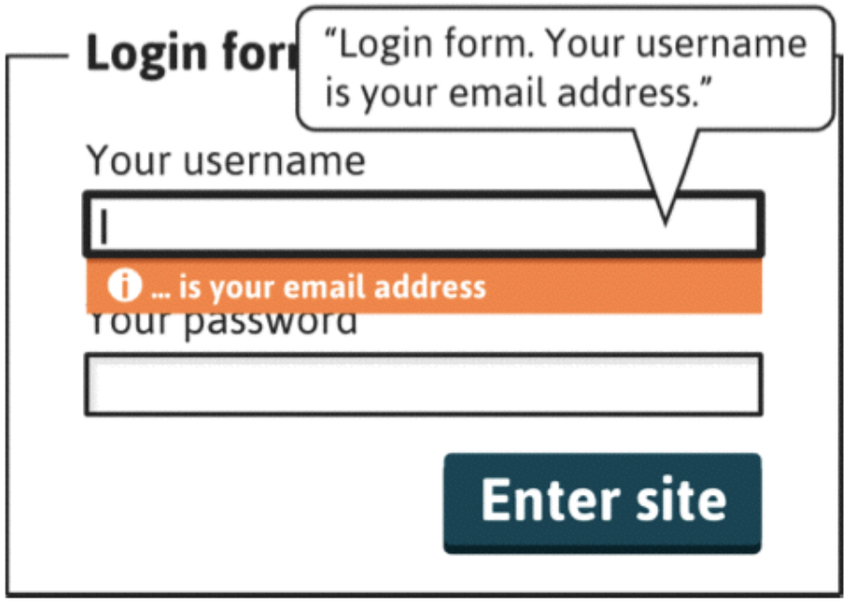
```
a.skip:focus {
    top: 0;
}
```

Accessibility Techniques

- Accessibility is about defining *relationships,* properties and behaviors of elements explicitly.
  - *Buttons* have different states; they control elements,
  - *Navigation* requires appropriate sectioning.
  - *Forms*
  - *Widgets*
- *Problem:*

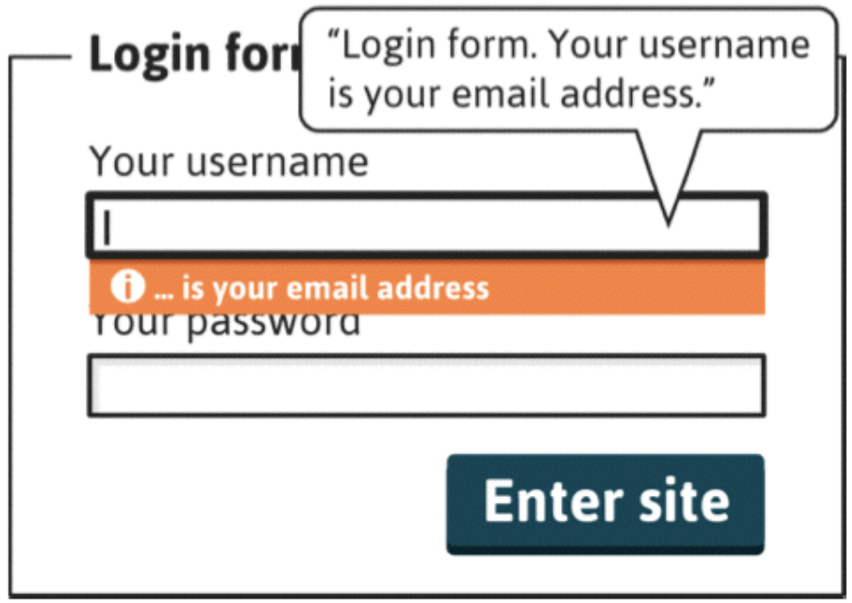
screen readers/keyboards—and


Login form
"Login form. Your username is your email address."
Your username
... is your email address
Your password
Enter site



- HTML:

```
<form>
    <fieldset><legend>Login form</legend>
        <div>
            <label for="username">Your username</label>
            <input id="username" type="text"
                   aria-describedby="username-hint">
            <div role="tooltip" id="username-hint">
                &hellip; is your email address</div>
        </div>
    </fieldset>
   <button type="submit">Enter site</button>
</form>
```



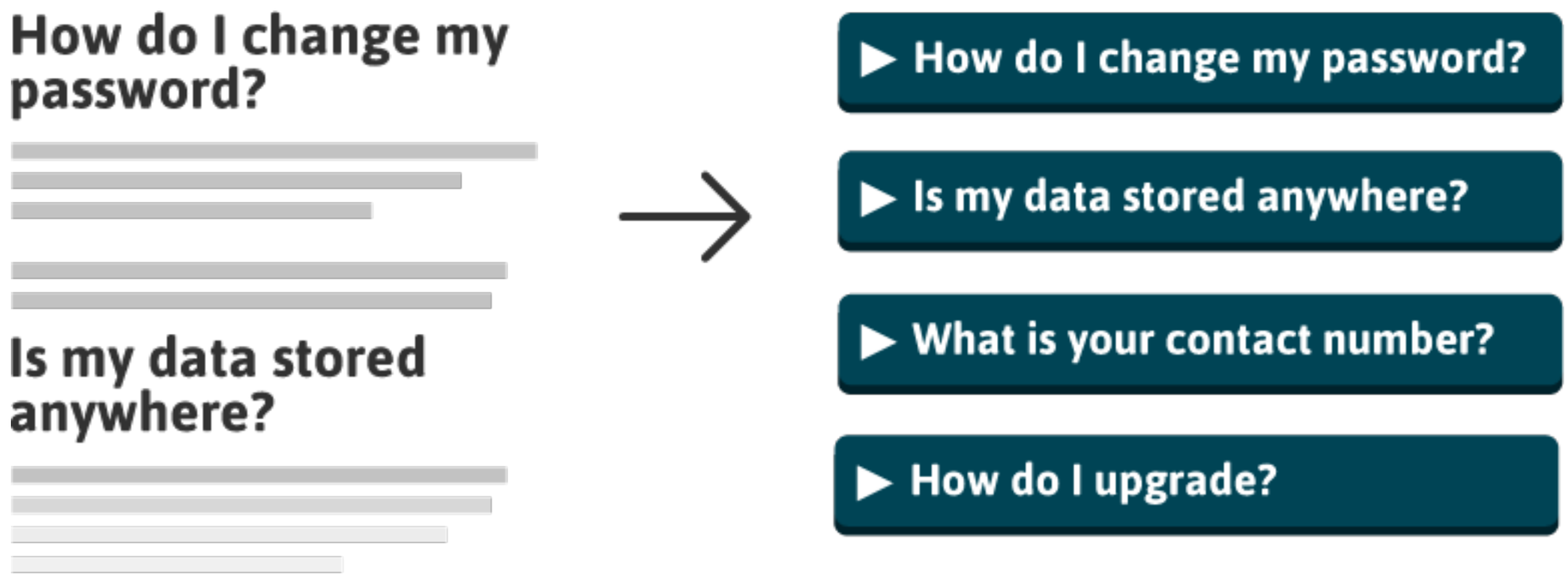
- CSS:

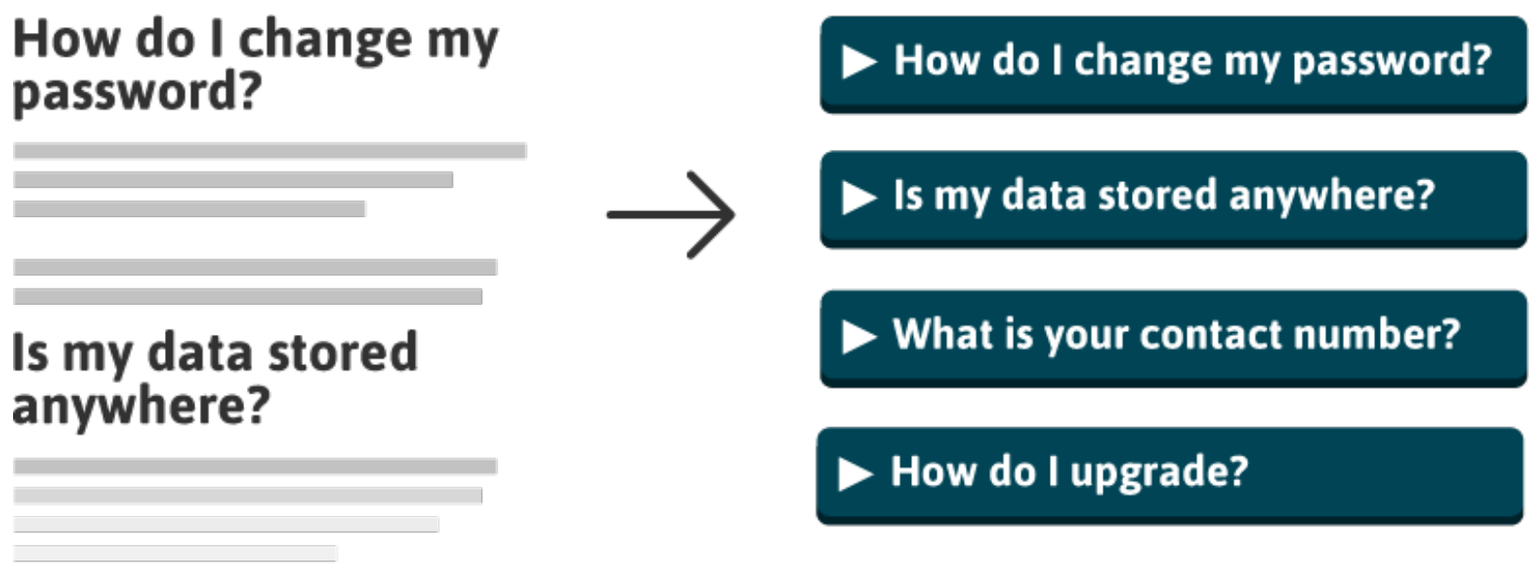
```
[role="tooltip"] {
    background: orange;
    color: white;
    padding: 0.5em;
    display: none;
}

input:focus + [role="tooltip"] {
    display: block;
}
```

Accessibility Techniques

- Accessibility is about defining *relationships,* properties and behaviors of elements explicitly.
  - *Buttons* have different states; they control elements,
  - *Navigation* requires appropriate sectioning.
  - *Forms* require concise and appropriate feedback,
  - *Widgets*
- *Problem:*

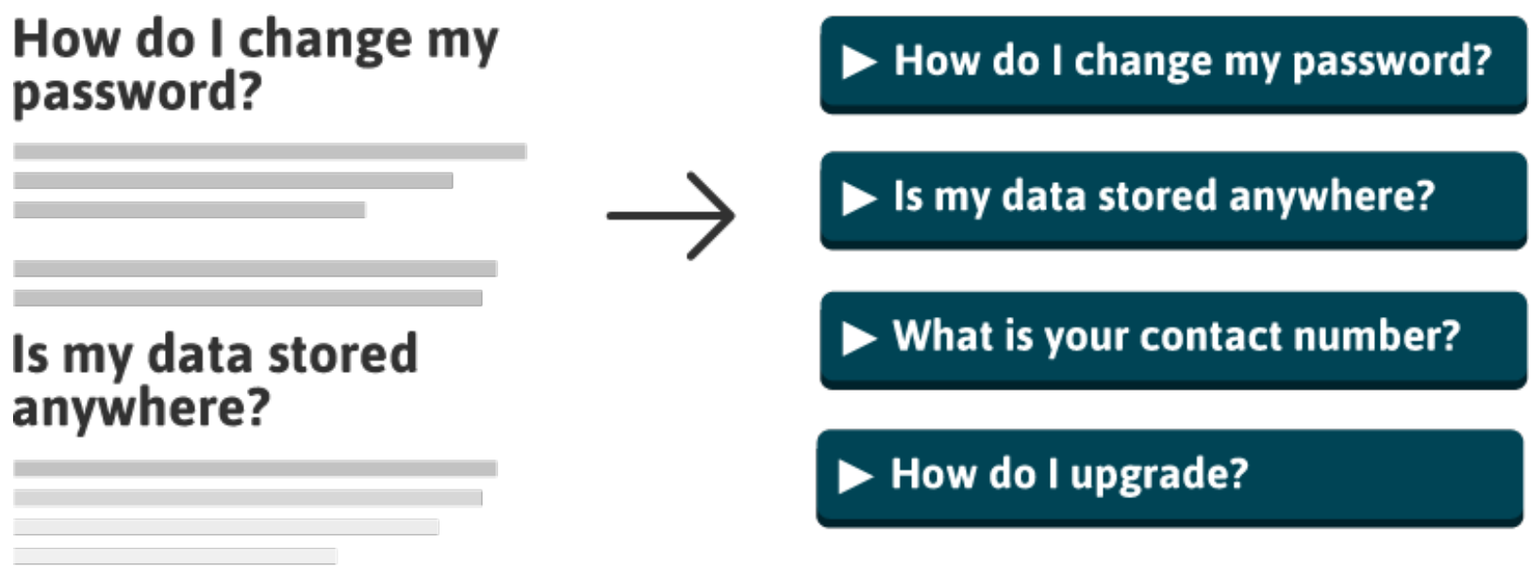
screen readers/keyboards—and


How do I change my password?
Is my data stored anywhere?
How do I change my password?
Is my data stored anywhere?
What is your contact number?
How do I upgrade?



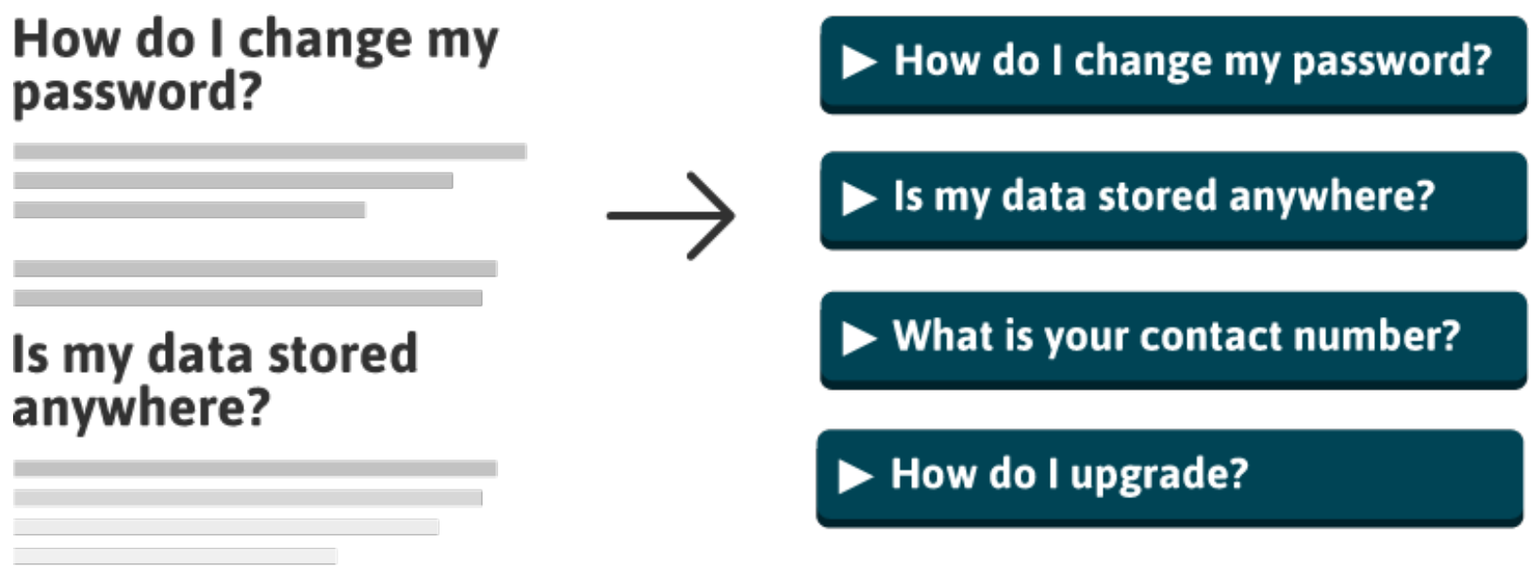
- HTML:

```
<h2>How do I change my password?</h2>
<p>…</p>

<h2>Is my data stored anywhere?</h2>
<p>…</p>

<h2>What is your contact number?</h2>
<p>…</p>
```



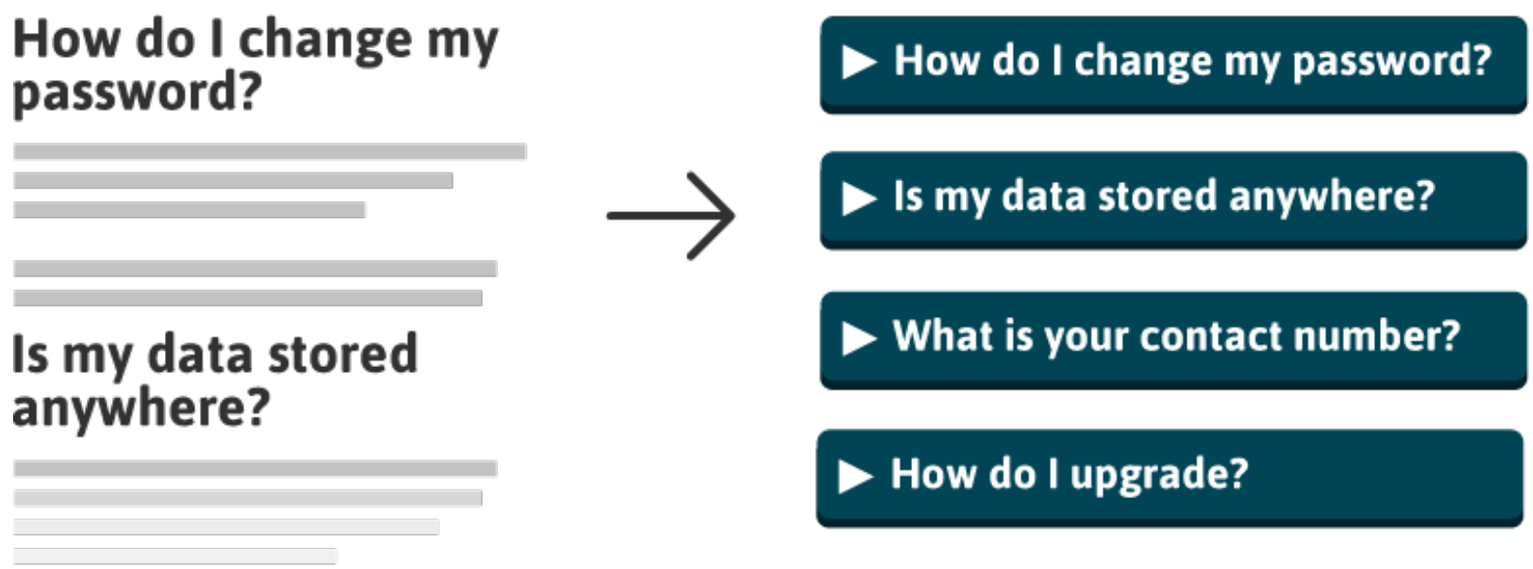
- HTML:

```
<h2 role="button">How do I change my password?</h2>
<p>…</p>

<h2 role="button">Is my data stored anywhere?</h2>
<p>…</p>

<h2 role="button">What is your contact number?</h2>
<p>…</p>
```



- HTML:

```
<h2>
    <button aria-expanded="false"
            aria-controls="change-pass">
            How do I change my password?
    </button>
</h2>
<div id="change-pass" aria-hidden="true">
    <p>…</p>
</div>
```



- CSS:

```
[aria-hidden] {
    display: none;
}

[aria-hidden="false"] {
    display: block;
}
```

```
[aria-expanded]:before {
    content: '\25ba\0020';
}

[aria-expanded="false"]:before {
    content: '\25bc\0020';
}
```

"Heading level 2. How do I change my password button. Collapsed."

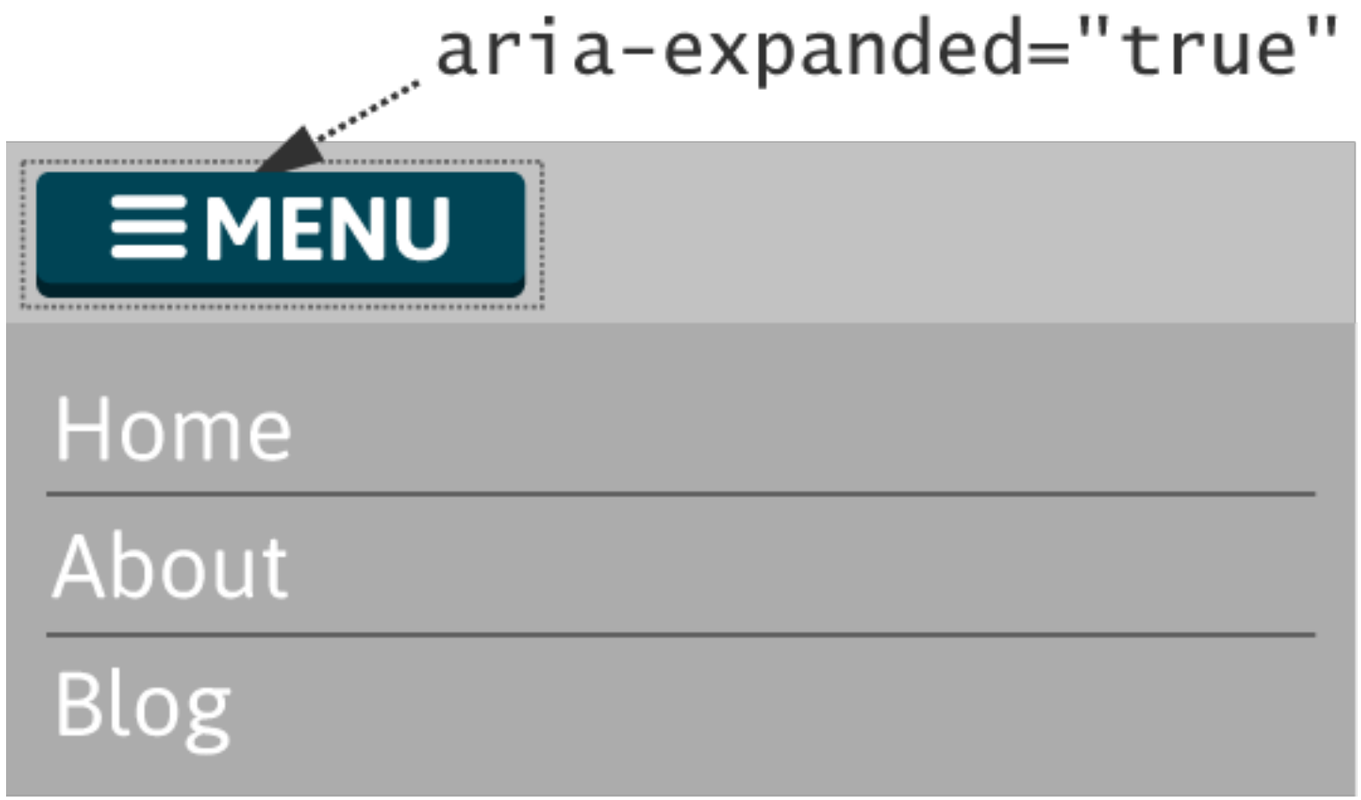
▶ How do I change my password?

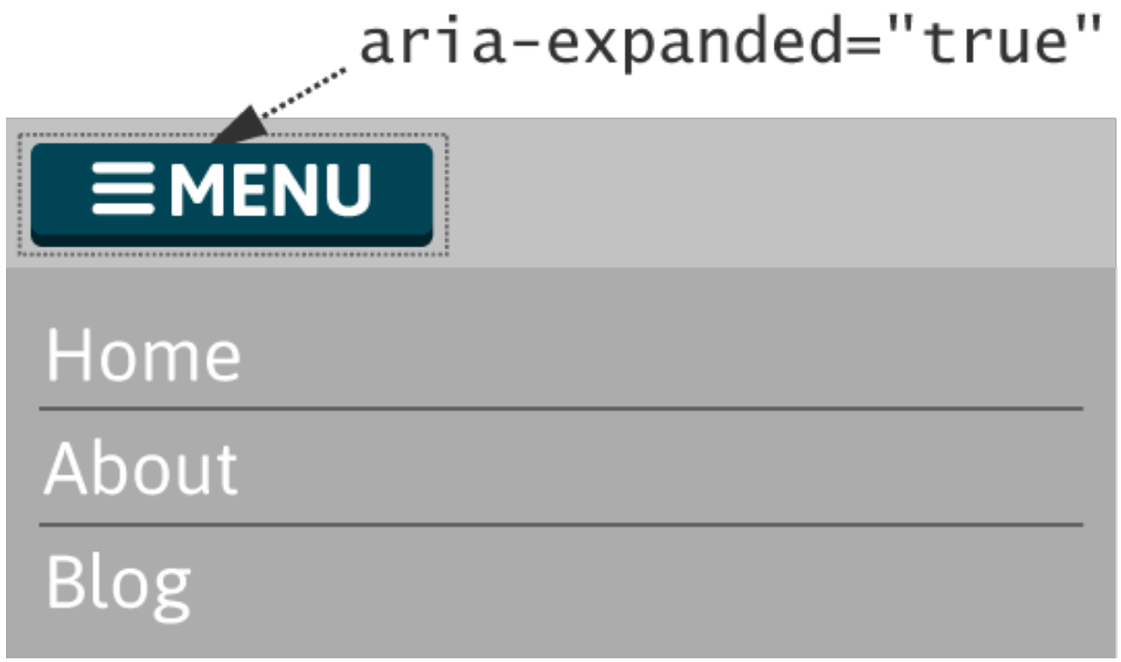
▶ Is my data stored anywhere?

▶ What is your contact number?

"There are seventeen steps to..."

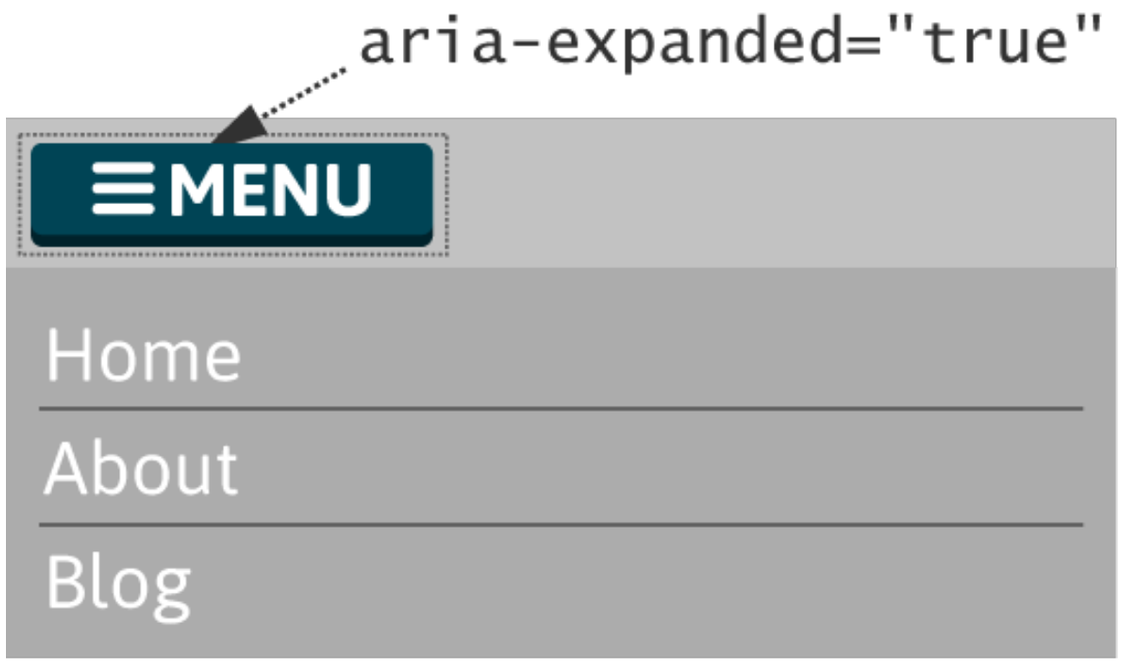
▼ How do change my password?


aria-expanded="true"
MENU
Home
About
Blog



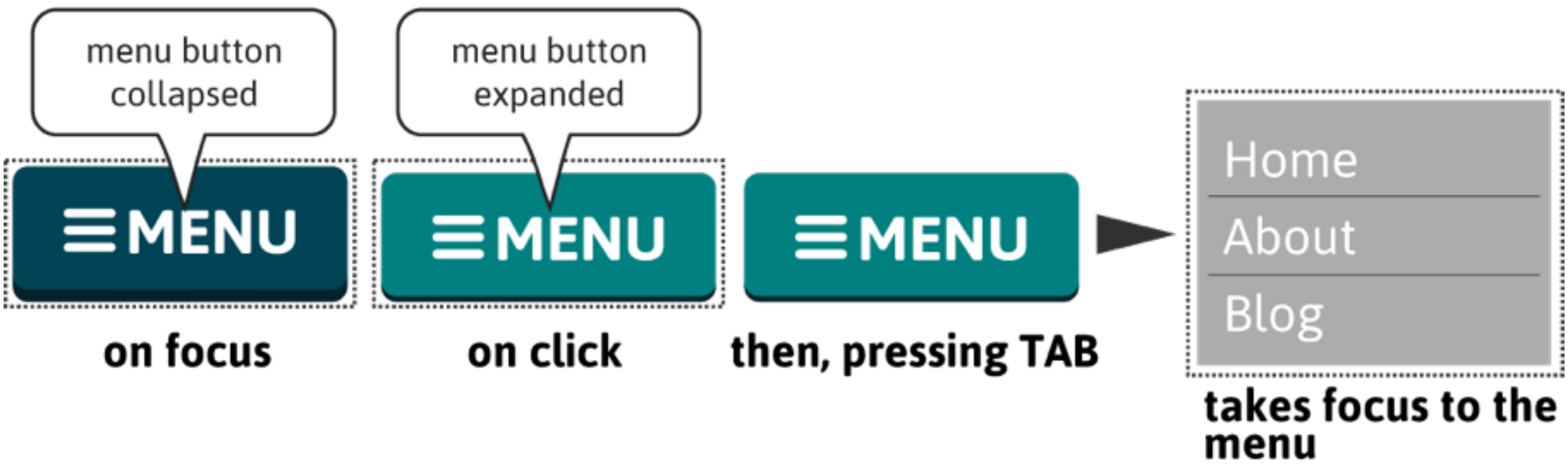
- HTML:

```html
<a href="#nav">Menu</a>
<!-- other markup… -->
<nav id="nav" role="navigation">
    <ul>
        <li><a href="/">Home</a></li>
        <li><a href="/about">About</a></li>
        <li><a href="/blog">Blog</a></li>
    </ul>
</nav>
```



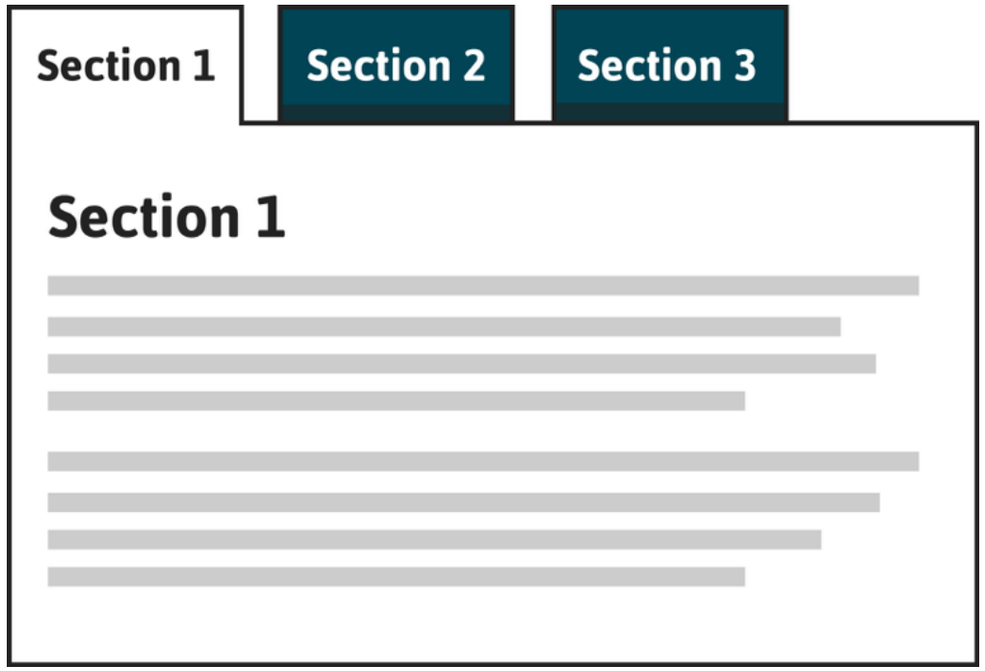
- HTML:

```
<a href="#nav" role="button" aria-controls="nav"
                             aria-expanded="false">Menu</a>
<!-- other markup… -->
<nav id="nav" role="navigation" aria-hidden="true">
    <ul>
        <li><a href="/">Home</a></li>
        <li><a href="/about">About</a></li>
        <li><a href="/blog">Blog</a></li>
    </ul>
</nav>
```


menu button collapsed
menu button expanded
MENU
MENU
MENU
Home
About
Blog
on focus
on click
then, pressing TAB
takes focus to the menu

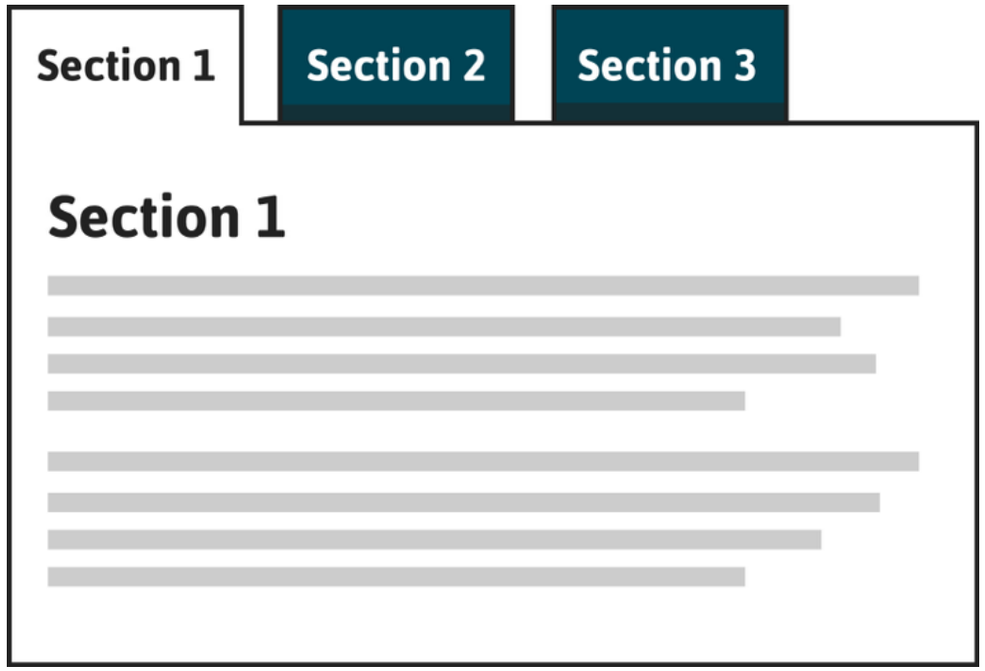
Section 1 | Section 2 | Section 3

Section 1



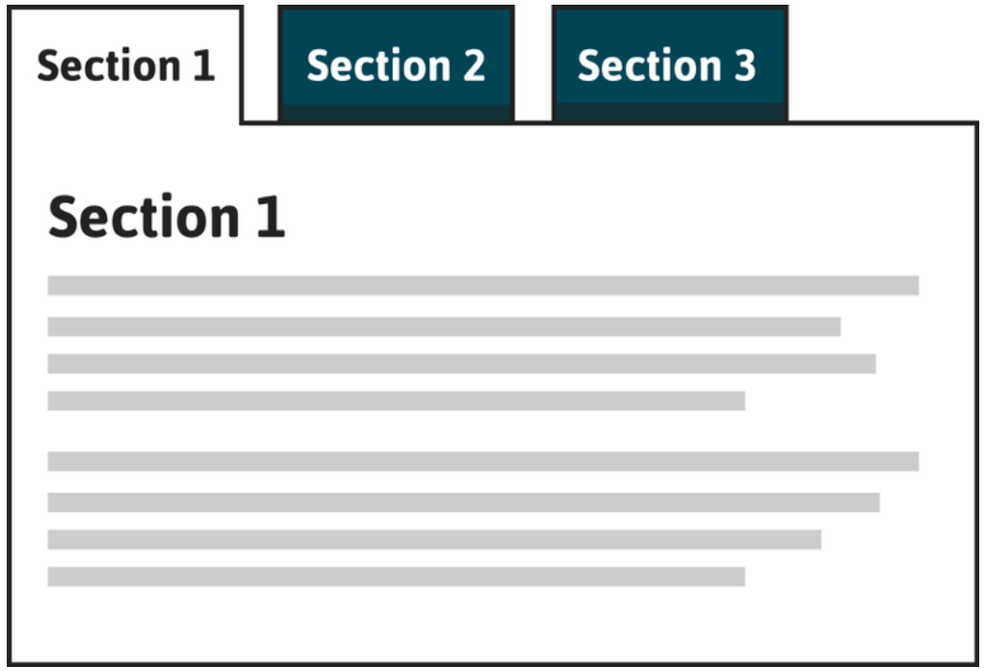
- HTML:

```html
<ul>
    <li><a href="#sec1">Section 1</a></li>
    <li><a href="#sec2">Section 2</a></li>
    <li><a href="#sec3">Section 3</a></li>
</ul>

<section id="sec1">...</section>
<section id="sec2">...</section>
<section id="sec3">...</section>
```



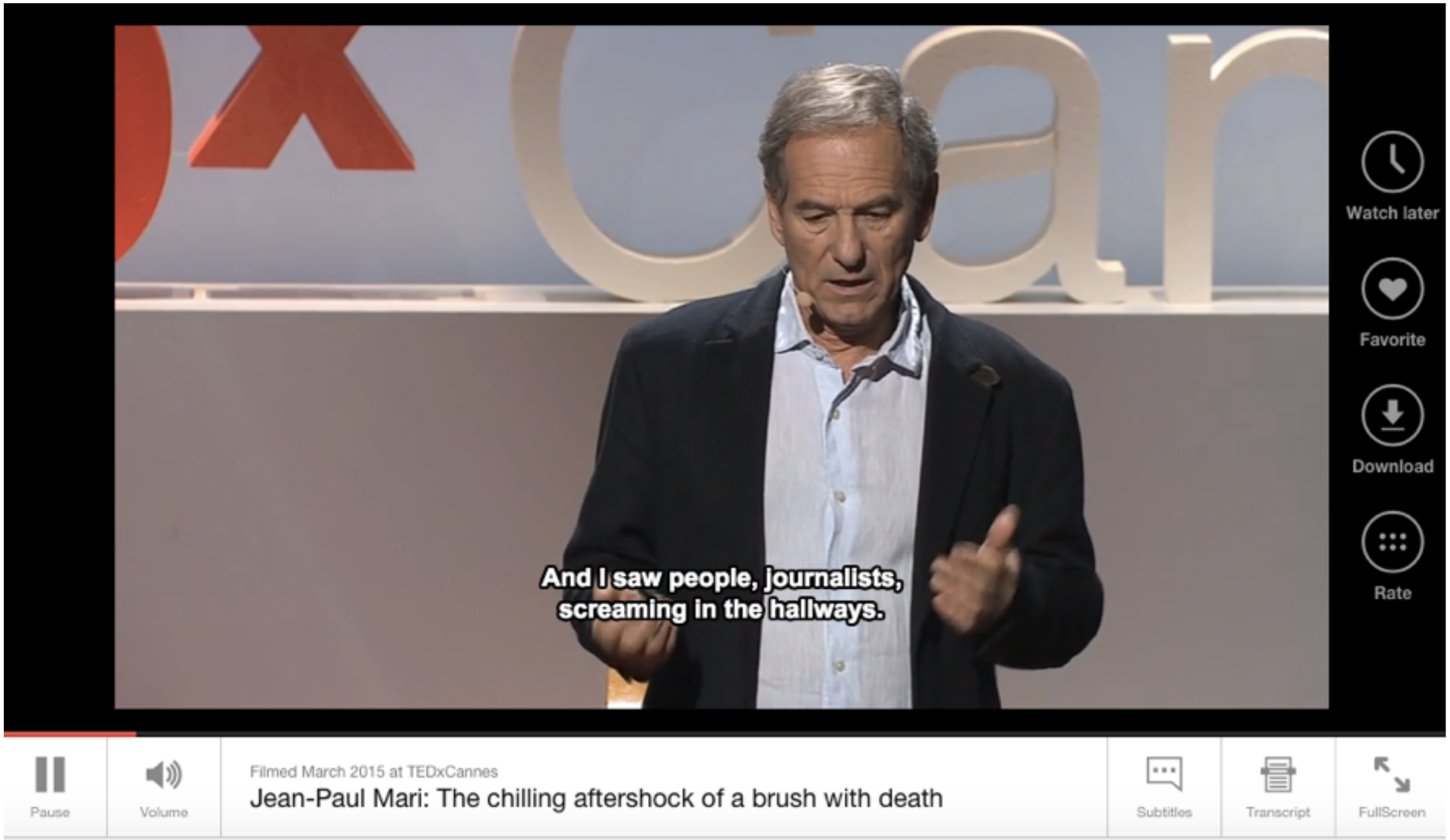
- HTML:

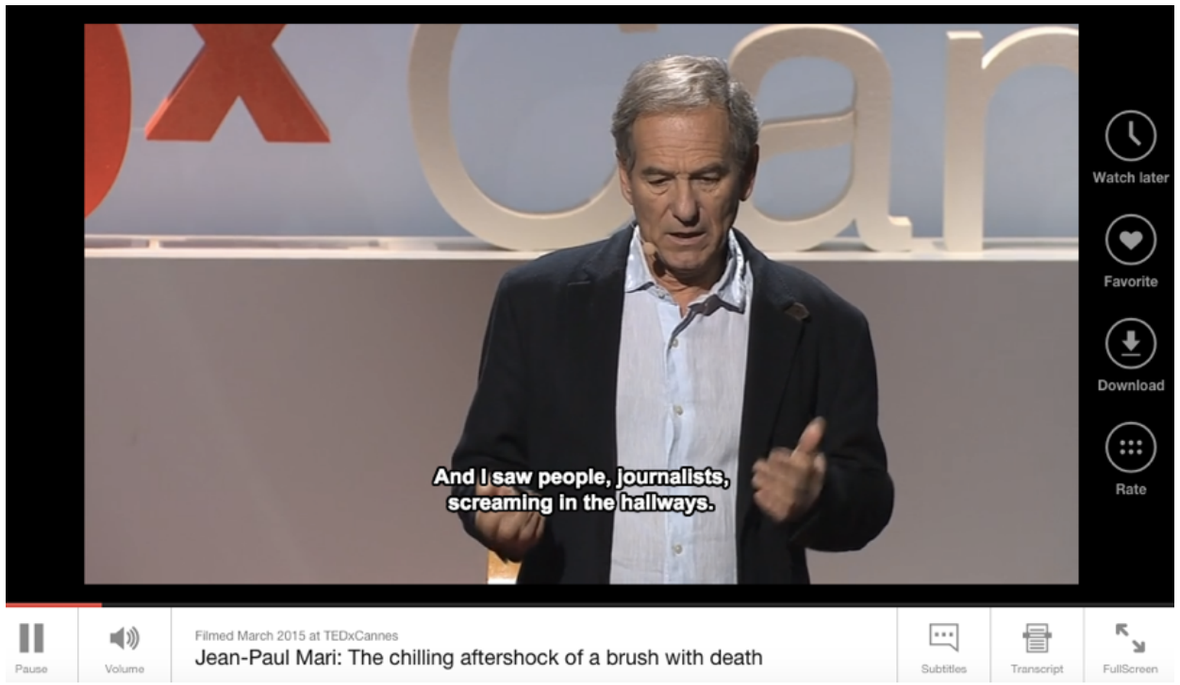
```
<ul role="tablist">
    <li><a role="tab" href="#sec1">Section 1</a></li>
    <li><a role="tab" href="#sec2">Section 2</a></li>
    <li><a role="tab" href="#sec3">Section 3</a></li>
</ul>

<section id="sec1" role="tabpanel">...</section>
<section id="sec2" role="tabpanel">...</section>
<section id="sec3" role="tabpanel">...</section>
```



- HTML:

```
<ul role="tablist">
    <li><a role="tab" aria-controls="panel1"
                      aria-selected="true"
           href="#sec1">Section 1</a></li>
    <li><a role="tab" aria-controls="panel2"
           href="#sec2">Section 2</a></li>
</ul>

<section id="sec1" role="tabpanel">...</section>
<section id="sec2" role="tabpanel" aria-hidden="true">...</section>
```

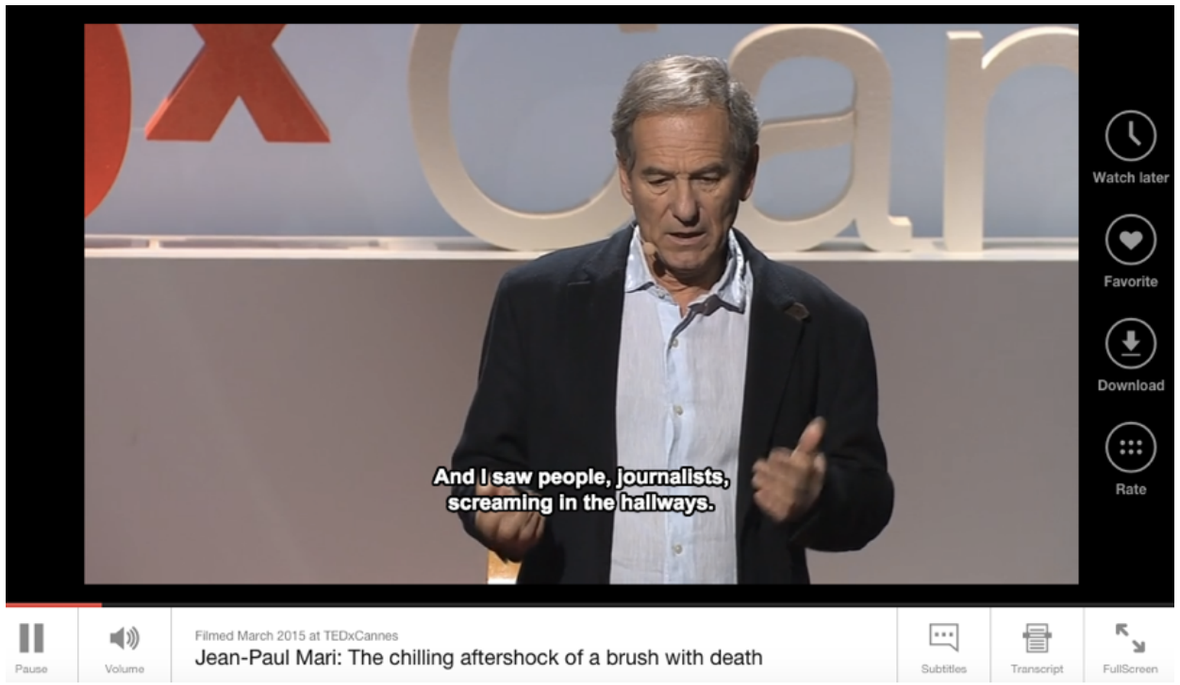

Watch later
Favorite
Download
Rate
And I saw people, journalists,
screaming in the hallways.
Pause
Volume
Filmed March 2015 at TEDxCannes
Jean-Paul Mari: The chilling aftershock of a brush with death
Subtitles
Transcript
FullScreen



- *Accessible video* includes captions, a transcript and an audio description.

- HTML:

```
<video preload="auto" width="480" height="360"
poster="myvideo.jpg">
    <source type="video/mp4" src="talk.mp4" />
    <source type="video/webm" src="talk.webm" />
    <track kind="captions" src="talk.vtt" />
</video>
```



- *Accessible video* includes captions, a transcript and an audio description.
- Delivered in a fully accessible video player, incl. *accessible controls* and playback options.

Able Player

ableplayer.github.io/ableplayer/

View on GitHub

Able Player

Fully accessible cross-browser HTML5 media player.

tar.gz .zip

Able Player

Able Player is a fully accessible cross-browser media player. It uses the HTML5 <audio> or <video> element for browsers that support them, and (optionally) the JW Player as a fallback for those that don't.

To see the player in action check our Able Player Examples page.

Features

- Supports both audio and video.
- Supports either a single audio track or an entire playlist.
- A full set of player controls that are keyboard-accessible, properly labeled for screen reader users, and controllable by speech recognition users.

paypal/accessible-html5-vi

Vitaly

GitHub, Inc. [US] https://github.com/paypal/accessible-html5-video-player

GitHub | This repository | Search | Explore | Features | Enterprise | Pricing | Sign up | Sign in

paypal / accessible-html5-video-player

Watch 137 | Star 1,594 | Fork 198

Accessible HTML5 Video Player http://paypal.github.io/accessible-html5-video-player/

52 commits | 2 branches | 0 releases | 6 contributors

Branch: master | accessible-html5-video-player / +

mpnkhan Merge pull request #54 from hozefaj/bower-component ··· Latest commit 6eaa927 18 days ago

css	fixing margin for reload button	26 days ago
images	adding fullscreen icons to sprite	5 months ago
js	Fixing issue #47 where chrome fails hasOwnProperty	3 months ago
media	Fixed typos	a year ago
HTML5 video notes.txt	copying project files	a year ago
LICENSE.md	copying project files	a year ago
README.md	adding link to related blog post	a year ago
bower.json	updating author	18 days ago
	aria-hidden on captions after testing with JAWS	a year ago

Code

Issues 13

Pull requests 2

Pulse

Graphs

HTTPS clone URL

https://github.com

You can clone with HTTPS or Subversion.

Clone in Desktop

Download ZIP

https://enterprise.github.com

Accessibility Techniques

- Accessibility is about defining *relationships,* properties and behaviors of elements explicitly.
  - *Buttons* have different states; they control elements,
  - *Navigation* requires appropriate sectioning.
  - *Forms* require concise and appropriate feedback,
  - *Widgets* have complex properties and behaviors,
- *Problem:*

screen readers/keyboards—and

Accessibility Strategy

- Accessibility is about defining *relationships,* properties and behaviors of elements explicitly.
  - Writing HTML, prefer *semantically sound elements,*
  - Define *explicit relationships* (roles, properties, states),
  - Check *global properties first,* then local properties.
  - Test *keyboard and screen readers* use.
  - Refine *accessibility enhancements.*

Patterns - The Accessibility

a11yproject.com/patterns/

A11Y Project | Patterns | Checklist | Resources | About

Patterns

An accessible widget & pattern library

Menus

- Dropdown
- Mega Dropdown
- Mobile Toggle
- Off Canvas Push

Imagery

- SVG
- Video Element Backgrounds

Accordions & Tabs

- Simple Accordion
- Simple Tabs
- Responsive Tabs to Accordion

Buttons

- Form Submission
- Single Toggle (anchor)

Anchors/Links

- Read More:
  - hidden
  - aria-labelledby
  - aria-label
- Skip To Content

Modal Windows

- The Incredible Accessible Modal Window, Version 3
- An accessible Modal Window with JavaScript & CSS

Tables

Tooltips

Video Players

Practical ARIA Examples

heydonworks.com/practical_aria_examples/

ARIA Examples

Some practical **ARIA**(?) examples to enhance your application accessibility (by @heydonworks).

1. Accessible input tooltips with no javascript
2. Button-controlled input with feedback
3. Progressive collapsibles
4. Simple ARIA tab interface
5. Alert! You're offline
6. Warning dialog (modal)
7. A simple toolbar widget
8. Progressive hamburger (off-canvas navigation)

Rindfleischetikettierungsüberwachungsaufgabenübertragungsgesetz

The word - which refers to the "law for the delegation of monitoring beef labelling", has been repealed by a regional parliament after the EU lifted a recommendation to carry out BSE tests on healthy cattle. German is famous for its compound nouns, which frequently become so cumbersome they have to be reduced to abbreviations. The beef labelling law, introduced in 1999 to protect consumers from BSE, was commonly transcribed as the "RkReÜAÜG", but even everyday words are shortened to initials so Lastkraftwagen - lorry - becomes Lkw.

Rindfleischetikettie-rungsüberwachungsa-ufgabenübertragungsge-setz

The word - which refers to the "law for the delegation of monitoring beef labelling", has been repealed by a regional parliament after the EU lifted a recommendation to carry out BSE tests on healthy cattle. German is famous for its compound nouns, which frequently become so cumbersome they have to be reduced to abbreviations. The beef labelling law, introduced in 1999 to protect consumers from BSE, was commonly transcribed as the "RkReÜAÜG", but even everyday words are shortened to initials so Lastkraftwagen - lorry - becomes Lkw.

Rindfleischetikettierungsüberwachungsaufgabenübertragungsgesetz

The word - which refers to the "law for the delegation of monitoring beef labelling", has been repealed by a regional parliament after the EU lifted a recommendation to carry out BSE tests on healthy cattle. German is famous for its compound nouns, which frequently become so cumbersome they have to be reduced to abbreviations. The beef labelling law, introduced in 1999 to protect consumers from BSE, was commonly transcribed as the "RkReÜAÜG", but even everyday words are shortened to initials so Lastkraftwagen - lorry - becomes Lkw.

Rindfleischetikettierun…

The word - which refers to the "law for the delegation of monitoring beef labelling", has been repealed by a regional parliament after the EU lifted a recommendation to carry out BSE tests on healthy cattle. German is famous for its compound nouns, which frequently become so cumbersome they have to be reduced to abbreviations. The beef labelling law, introduced in 1999 to protect consumers from BSE, was commonly transcribed as the "RkReÜAÜG", but even everyday words are shortened to initials so Lastkraftwagen - lorry - becomes Lkw.

Conclusion

I tested all the examples above and combinations of them in IE7, IE8, IE9, IE10, IE11, Edge, Firefox 39 (Windows, Linux, Mac), Chrome 44 (Windows, Linux, Mac), Opera 30 (Windows, Mac), Safari 8 (Mac), Safari 5.1 (Windows), Android 5 (Nexus 6), Android 4.4 (Nexus 5), Android 2.3 (Galaxy S2), iOS 8.3 (iPhone 6), iOS 7 (iPhone 5S), iOS 6 (iPhone5), Opera Mini (Android 5), Opera Classic (Android 5), Opera Mobile (Android 5) and Windows Phone 8.1 (Lumia 930) using real devices and BrowserStack – Here is a list of all 26 browsers and their results.

When searching the web you probably will find the following solution:

```
.hyphenate {
  -ms-word-break: break-all;
  word-break: break-all;
  word-break: break-word;

  -webkit-hyphens: auto;
  -moz-hyphens: auto;
  hyphens: auto;
}
```

While this works great in most cases, I found out that in Firefox hyphens won't work (altough

Dealing with long words in

https://justmarkup.com/log/2015/07/31/dealing-with-long-words-in-css/

Final solution

```
.hyphenate {
  overflow-wrap: break-word;
  word-wrap: break-word;
  -webkit-hyphens: auto;
  -ms-hyphens: auto;
  -moz-hyphens: auto;
  hyphens: auto;
}
```

This solution will show hyphens for every browser supporting it and will break lines in every other browser – perfect. Altough I have tested this solution in 26 different browsers I am still not sure this will work 100% – if you find any edge case please let me know.

Update 28.09.2015

Some people asked why I didn't mention HTML soft hyphens. There are two reasons. First of all, this post is about solutions using CSS and not HTML and second and more important is that I think it's impractical to define possible hyphens in the text itself and almost impossible for editors to do that by hand.

Comments are closed.

Dealing with long words in C | Can I use... Support tables f

caniuse.com/#feat=wordwrap

CSS3 Overflow-wrap - WD

Global 59.14% + 38.39% = 97.53%

Allows lines to be broken within words if an otherwise unbreakable string is too long to fit. Currently mostly supported using the `word-wrap` property.

Current aligned | Usage relative | Show all

IE	Edge *	Firefox	Chrome	Safari	Opera	iOS Safari *	Opera Mini *	Android Browser *	Chrome for Android
8			31					4.1	
9		38	43					4.3	
10		39	44			7.1		4.4.4	
11	12	40	45	8	31	8.4	8	44	44
	13	41	46	9	32	9			
		42	47		33				
		43	48						

Notes | Known issues (0) | Resources (6) | Feedback

Partial support refers to requiring the legacy name "word-wrap" (rather than "overflow-wrap") to work.

Dealing with long words in C

https://justmarkup.com/log/2015/07/31/dealing-with-long-words-in-css/

JUST MARKUP

Blog About Photos Contact

Dealing with long words in CSS

July 31, 2015

The web consists of content, content consists of words and words can be long, very long. Everyone involved with the web will sooner or later have to deal with long words.

Rindfleischetikettierungsüberwachungsaufgabenübertragungsgesetz

The word - which refers to the "law for the delegation of monitoring beef labelling", has been repealed by a regional parliament after the EU lifted a recommendation to carry out BSE tests on healthy cattle. German is famous for its compound nouns, which frequently become so cumbersome they have to be reduced to abbreviations. The beef labelling law, introduced in 1999 to protect consumers from BSE, was commonly transcribed as the "RkReÜAÜG", but even everyday words are shortened to initials so Lastkraftwagen - lorry - becomes Lkw.

1-1	**1-2**	**1-3**	**1-4**
2-1 Example text goes here. Example text goes here.	2-2	2-3	2-4
3-1	3-2	3-3	3-4
4-1	4-2	4-3	4-4

1-1	1-2	1-3
2-1 Example text goes here. Example text goes here.	2-2 Example text	2-3
3-1	3-2	3-3
4-1	4-2	4-3

Default Tables are Weird

I'm a table	and stuff

I'm as wide	as the cells are

I can easily become 100% wide.	and I'll stop there if I possibly can.

If you don't specify it,	the cells will be kind of arbitrarily wide depending on the content inside. This one is way wider right? ¯\(°_o)/¯

If you **do** specify it,	that will be respected if it's possible to do so. These should be 50% wide.

But things can get weird.	I told both of these cells to be 1000px wide and it's like NOPE.

It's kinda like a calculation. I'm told to be 2000px wide, thus I'm 66.66% wide (2/3 of total 3000px).	And I'm told to be 1000px wide, thus I'm 33.33% wide (1/3 of total 3000px).

An image that's too wide though?	That'll bust out like nobody's business. You can try hiding the overflow but that doesn't work.

Or some non-wrapping text?	That'll blast out a table in short order like thi

But seriously, as always, a live example will help. In the following demo, the table has `table-layout: fixed` added in the CSS. Click the toggle button to toggle it off, then on, etc.

HTML CSS JS Result Edit on CODEPEN

Using the `table-layout` property

1-1	1-2	1-3	1-4
2-1 Example text goes her…	2-2 Example text	2-3	2-4
3-1	3-2	3-3	3-4
4-1	4-2	4-3	4-4

TOGGLE TABLE-LAYOUT: FIXED

You can see in this example the advantage of using `table-layout: fixed`, as opposed to the default of `auto`. This won't always be the best choice and it won't always be necessary, but it's a nice one to keep in mind when dealing with tables that have cells with variable-width data.

Chris Coyier did a great write-up on this property last year, so if you want a much more comprehensive discussion, that's your best bet.

> With *table-layout: fixed;* the layout is fixed based on the first row. Set the width of those, and the rest of the table follows.

— *Chris Coyier*

Fixed Table Layouts | CSS-

https://css-tricks.com/fixing-tables-long-strings/

With `table-layout: fixed;`

Things get a lot sturdier and more predictable with property/value in place.

The layout is fixed based on the first row. Set the width of those, and the rest of the table follows.

It's a little more complicated, but not much. Here's an exploration:

HTML CSS Result Edit

```
td, th {
  border: 1px solid black;
}
table {
  margin: 15px 0;
  border: 1px solid black;
  table-layout: fixed;
  width: 100%; /* must have this set */
}
html {
  background: #ccc;
}
body {
  max-width: 400px;
  margin: 0 auto;
  background: white;
  padding: 10px;
}

.sturdy td:nth-child(1),
```

A Pen by Chris Coyier

codepen.io/chriscoyier/pen/xFcrp

CODEPEN | Fork | Share | Settings | Editor

HTML | Tidy

```html
<table class="users">
  <thead>
    <tr>
      <th class="row-1 row-ID">ID</th>
      <th class="row-2 row-name">Name</th>
      <th class="row-3 row-job">Job</th>
```

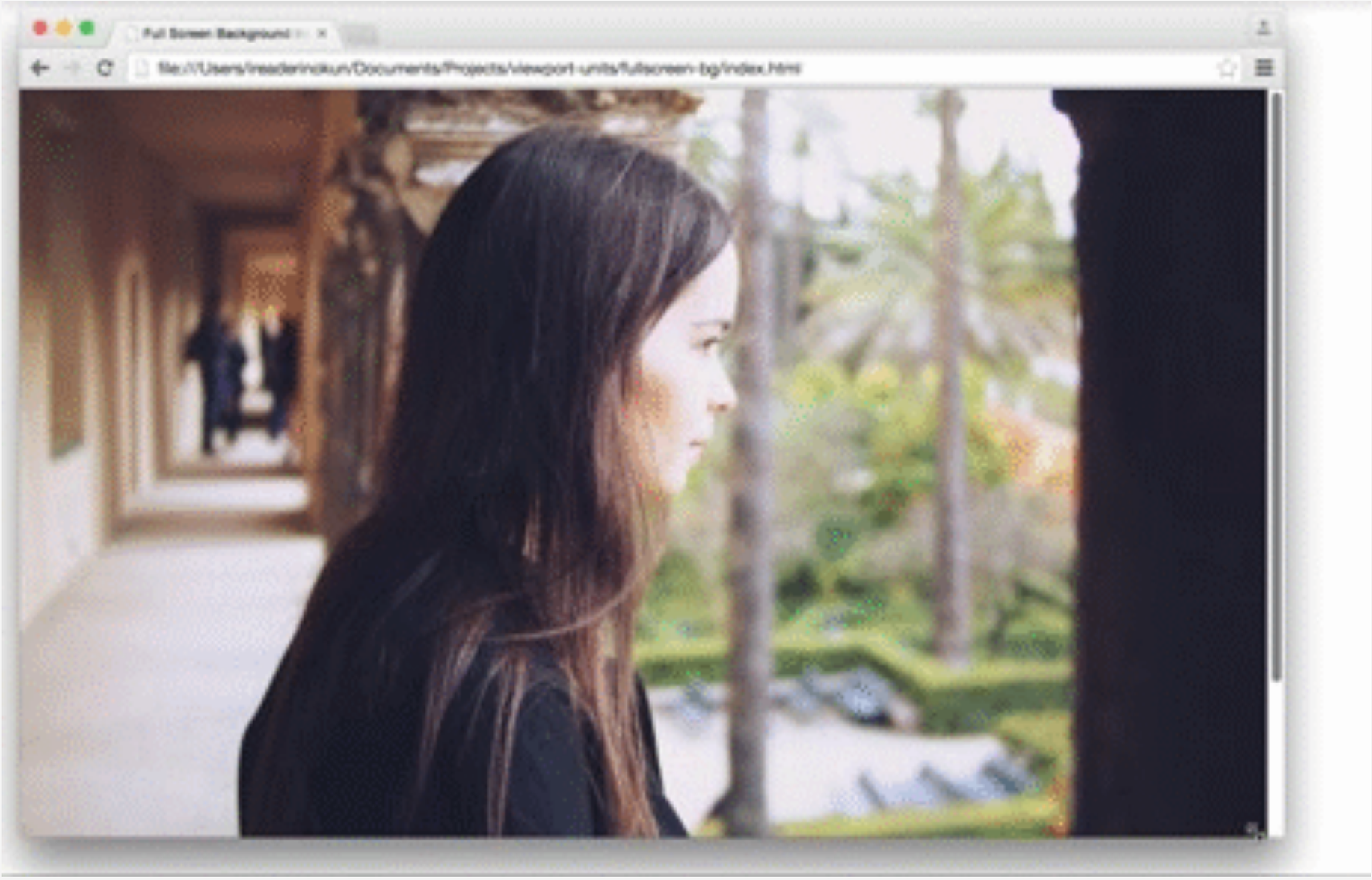
CSS | Tidy | View Compiled

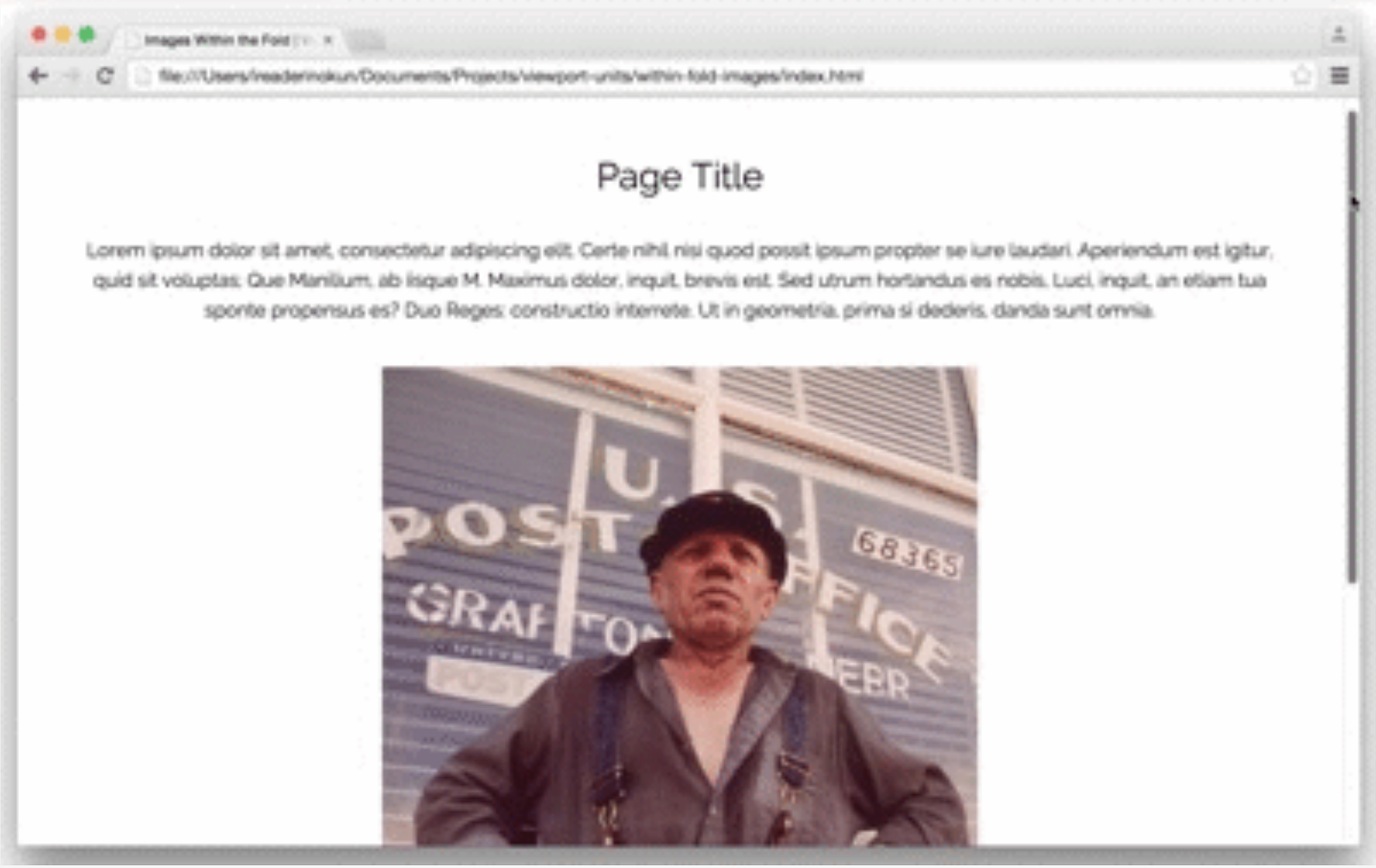
```css
.users td {
  white-space: nowrap;
  overflow: hidden;
  text-overflow: ellipsis;
}
.users th {
  background: darkblue;
  color: white;
}
```

JS

ID	Name	Job	Email
0001	Johnny Five	Robotin'	need@input.com
0002	Super Superlonglastnamesmith	Doin' stuff	doing@stuff.com
0003	Roger Wilco	Truckdrivin'	roger@wilco.com
0004	Mad Hatter	Hat Makin'	loves@mercury....

CodePen PRO is pretty sweet. Just saying. | Embed | Details | ⌘ | A Pen by Chris Coyier | HTML | CSS | JS


Full Screen Background
file:///Users/readerinckun/Documents/Projects/viewport-units/fullscreen-bg/index.html


Images Within the Fold
file:///Users/readerinokun/Documents/Projects/viewport-units/within-fold-images/index.html
Page Title
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Certe nihil nisi quod possit ipsum propter se iure laudari. Aperiendum est igitur, quid sit voluptas. Que Manilium, ab isque M. Maximus dolor, inquit, brevis est. Sed utrum hortandus es nobis, Luci, inquit, an etiam tua sponte propensus es? Duo Reges: constructio interrete. Ut in geometria, prima si dederis, danda sunt omnia.
68365

Viewport vs Percentage Un

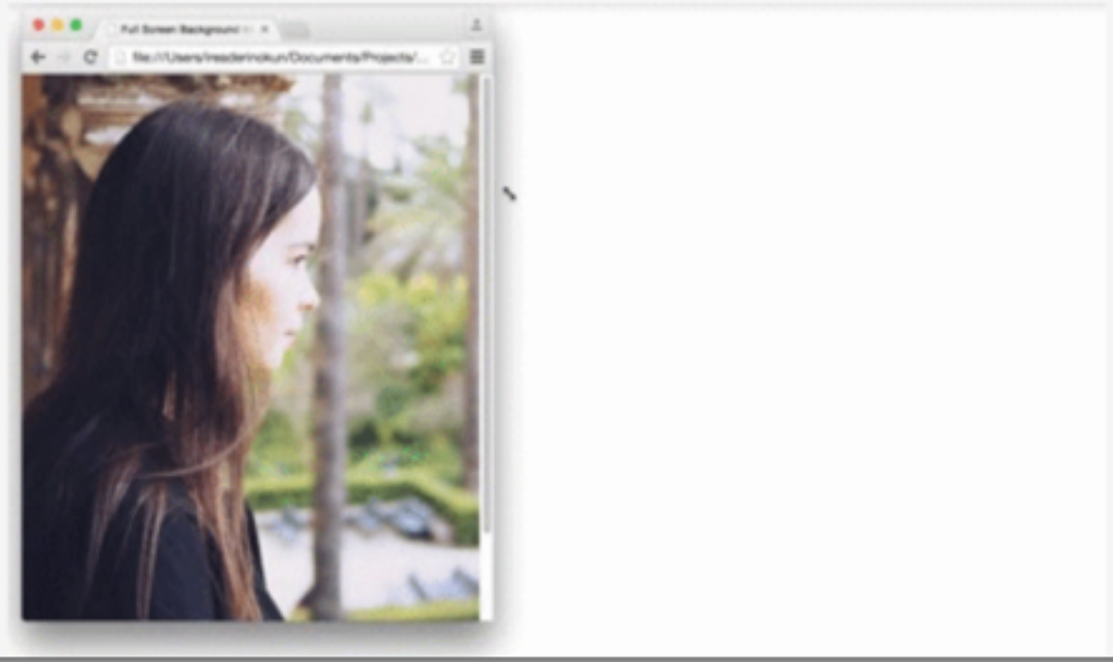
bitsofco.de/2015/viewport-vs-percentage-units/

Vitaly

THE FULLSCREEN BACKGROUND IMAGE

A typical use for the vh unit is for creating a background image that spans the entire height and width of the screen, no matter the size of the device. This is easily created with vh -

```
.bg {
  position: relative;
  background: url('bg.jpg') center/cover;
  width: 100%;
  height: 100vh;
}
```



Viewport vs Percentage Un

bitsofco.de/2015/viewport-vs-percentage-units/

WITHIN THE FOLD IMAGES

The vh unit can also be used to control the size of images within the page. For example, within an article, we may want to make sure that any image can be viewed in its entirety on the page, no matter the size of the screen.

To do that, we could set -

```
img {
  width: auto; /* Image width adjust to height to remain proportional */
  max-width: 100%; /* Image doesn't exceed parent element's width */
  max-height: 90vh; /* Image doesn't exceed viewport height */

  margin: 2rem auto;
}
```

Can I use... Support tables

caniuse.com/#search=vh

Viewport units: vw, vh, vmin, vmax - CR

Global 67.51% + 10.6% = 78.11%

Length units representing 1% of the viewport size for viewport width (vw), height (vh), the smaller of the two (vmin), or the larger of the two (vmax).

Current aligned | Usage relative | Show all

IE	Edge *	Firefox	Chrome	Safari	Opera	iOS Safari *	Opera Mini *	Android Browser *	Chrome for Android
8		38	31					4.1	
[1] 9		39	43			[3] 7.1		4.3	
[2] 10		40	44		31	8.4		4.4.4	
[2] 11	[2] 12	41	45	8	32	9	8	44	44
	[2] 13	42	46	9	33				
		43	47		34				
		44	48						

Notes | Known issues (10) | Resources (7) | Feedback

[1] Partial support in IE9 refers to supporting "vm" instead of "vmin".

[2] Partial support refers to not supporting the "vmax" unit.

Mixing vw and vh in font-si

codepen.io/CrocoDillon/pen/fBJxu

CODEPEN | Fork | Share | Settings | Editor | Log In | Si

HTML | Tidy

```
<p>Using calc in combination with vw and vh units for font-size to create text that always fills the viewport. No matter what ratio.
</p>
```

CSS | Tidy

```
p {
  margin: 0;
  font-size: calc(4vw + 4vh + 2vmin);
  /* See:
   * http://codepen.io/CrocoDillon/pen/jgmwt
   * For some math behind this
   */
```

JS

Using calc in combination with vw and vh units for font-size to create text that always fills the viewport. No matter what ratio.

CodePen PRO is pretty sweet. Just saying. | Embed | Details | ⌘ | A Pen by CrocoDillon | HTML | CSS | JS