Open UI

Customizable Control UI: Solving a Multi-Decade Problem

Greg Whitworth (@gregwhitworth) - Salesforce
Melanie Richards (@soMelanieSaid) - Microsoft

Agenda

Presentation (1 hour)

- **Problem**
- **What is a Control?**
- **Spectrum of Customization**
- **Solving Fully Style-Able Controls**
- **Solving Fully-Extensible Controls**
- **Process**

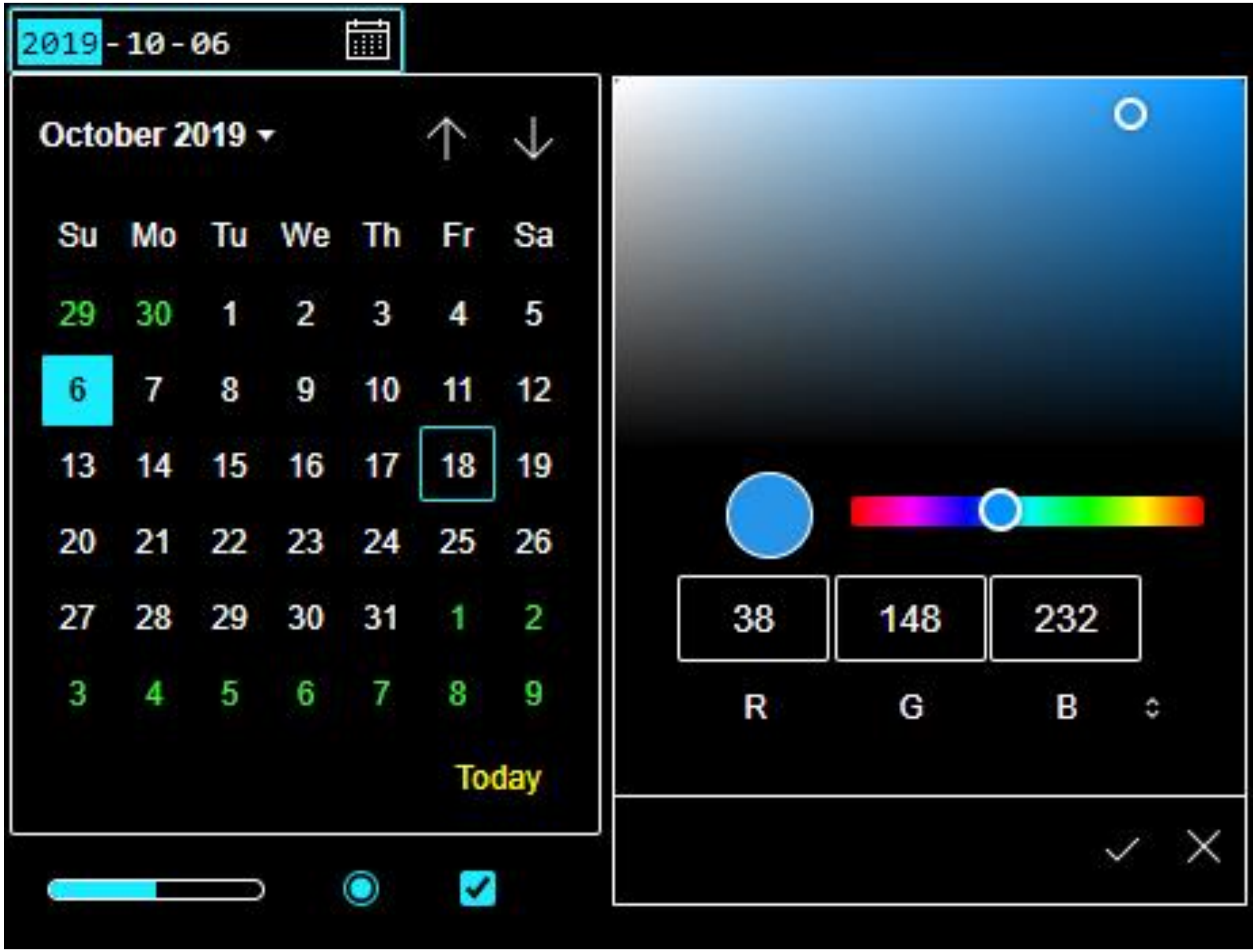
Break (15 min)

Discussion + Resolutions (1 hour)

Break (15 min)

Discussion + Resolutions (1.5 hours)

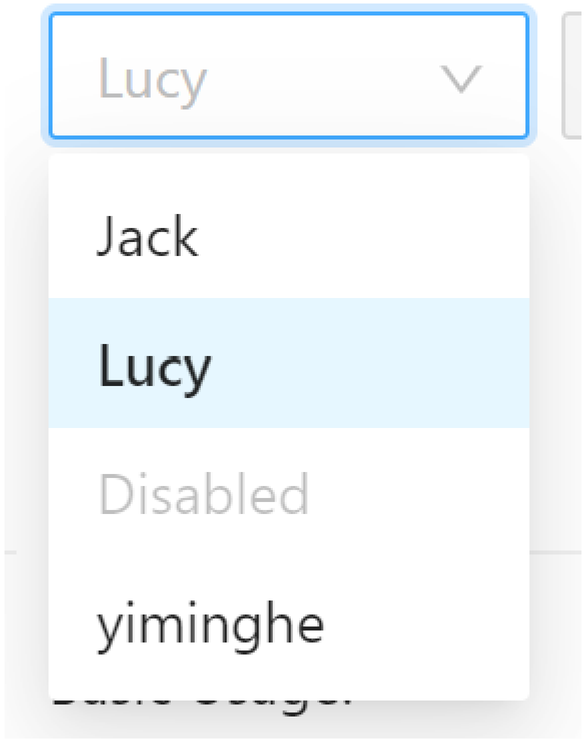
Problem

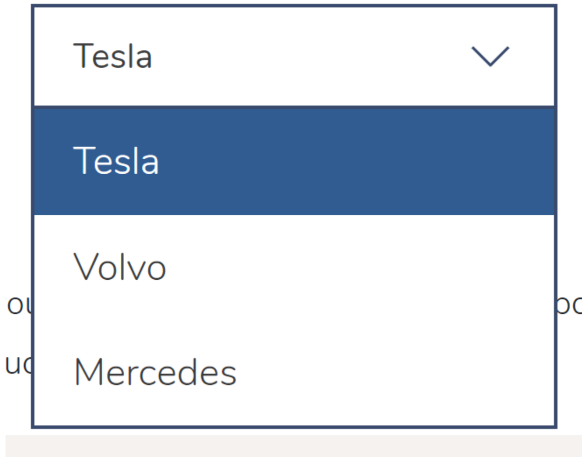

2019-10-06

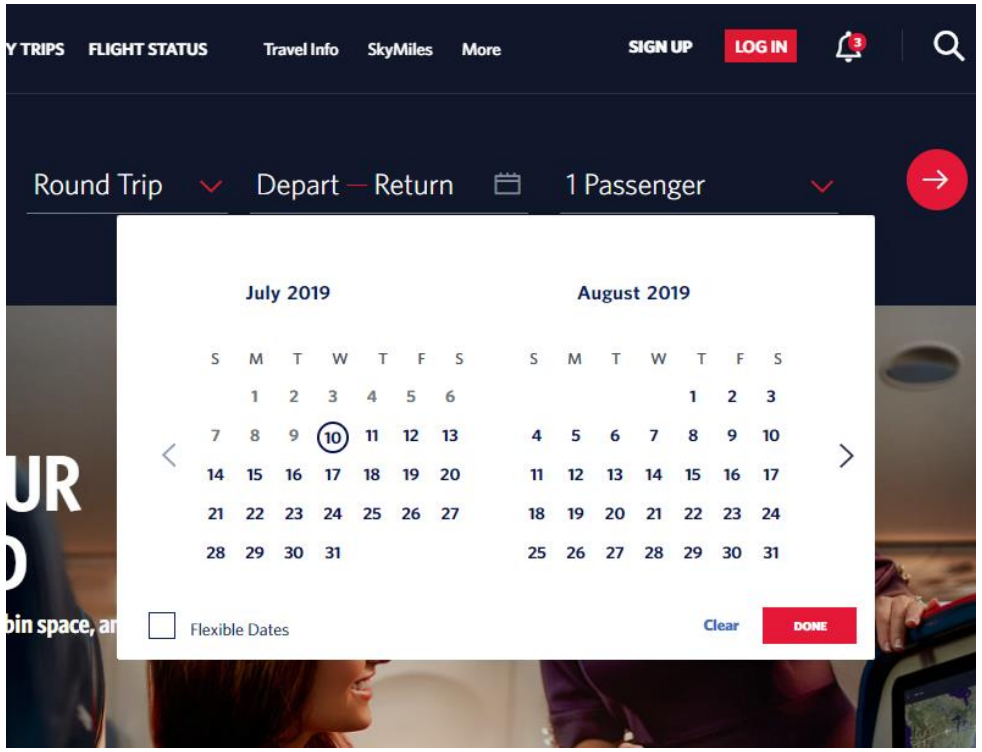
October 2019

Su	Mo	Tu	We	Th	Fr	Sa
29	30	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	31	1	2
3	4	5	6	7	8	9

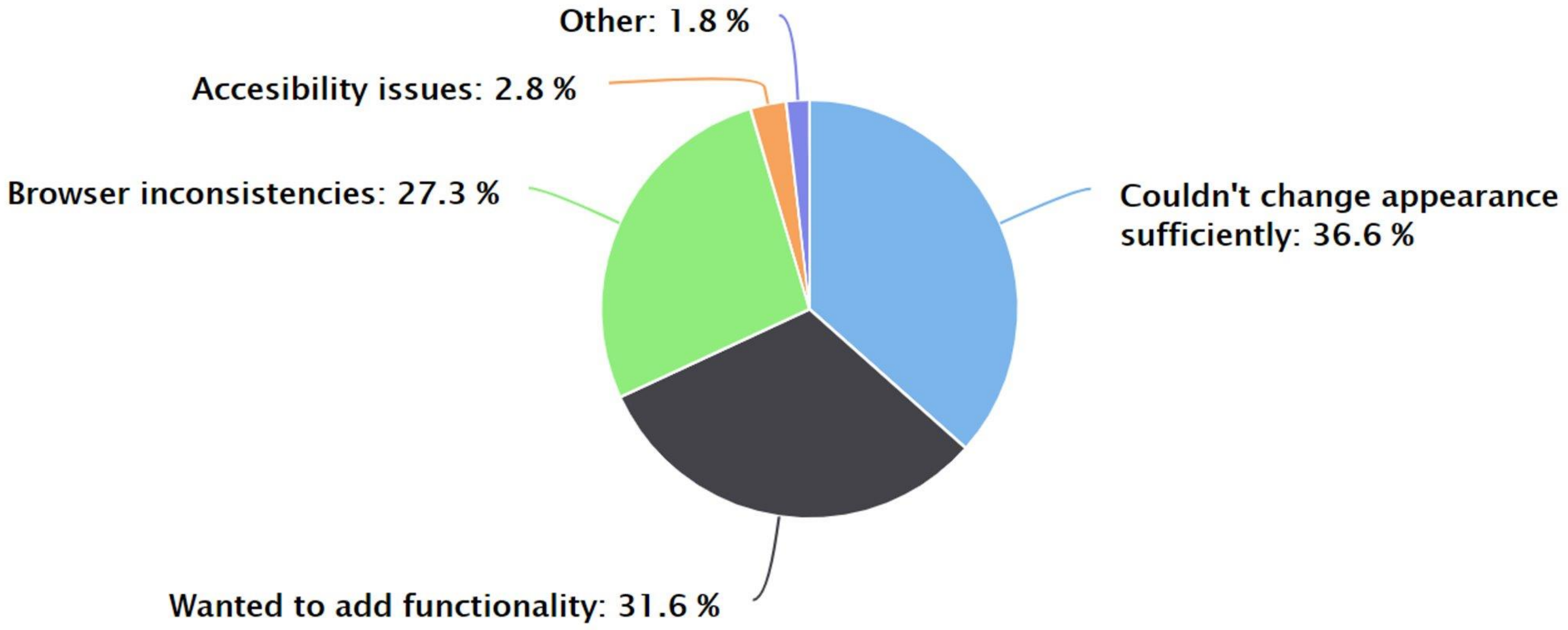
R: 38
G: 148
B: 232


Lucy
Jack
Lucy
Disabled
yiminghe


Tesla
Tesla
Volvo
Mercedes

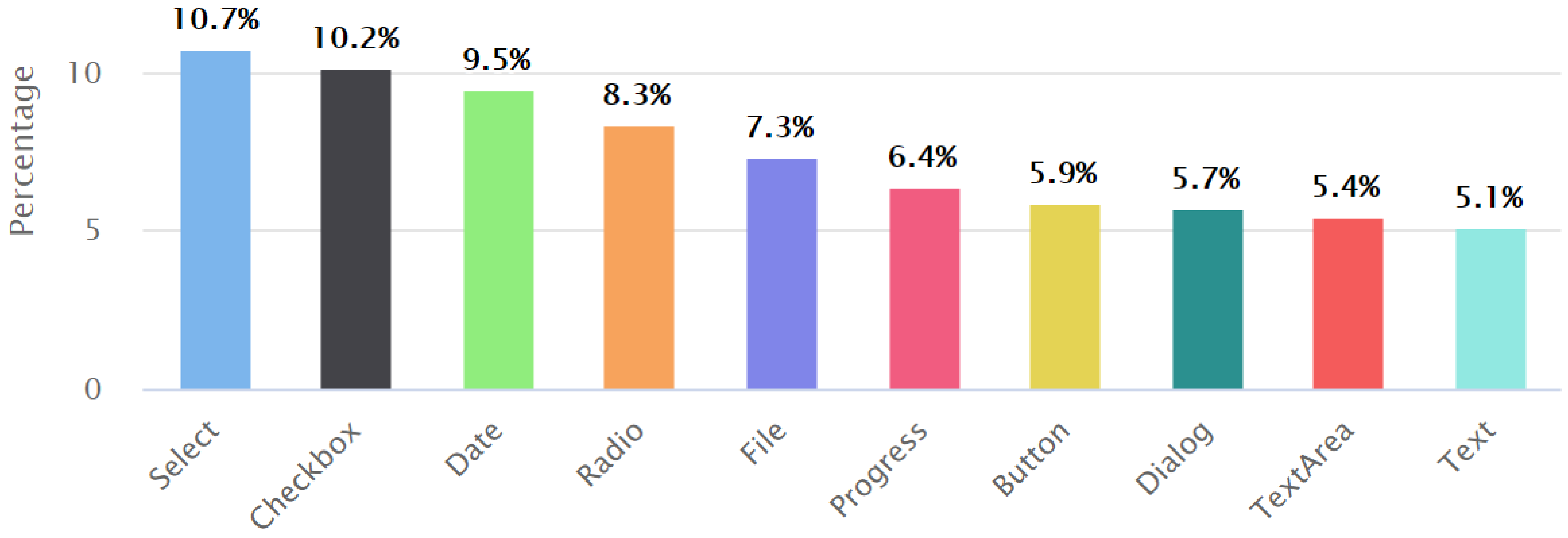

Y TRIPS
FLIGHT STATUS
Travel Info
SkyMiles
More
SIGN UP
LOG IN
Round Trip
Depart — Return
1 Passenger
July 2019
August 2019
S M T W T F S
Flexible Dates
Clear
DONE

Survey results: why developers recreate form controls



Sou

Survey results: top 10 controls recreated by web developers



Source: gwhitworth.com

FAST DNA

Fabric

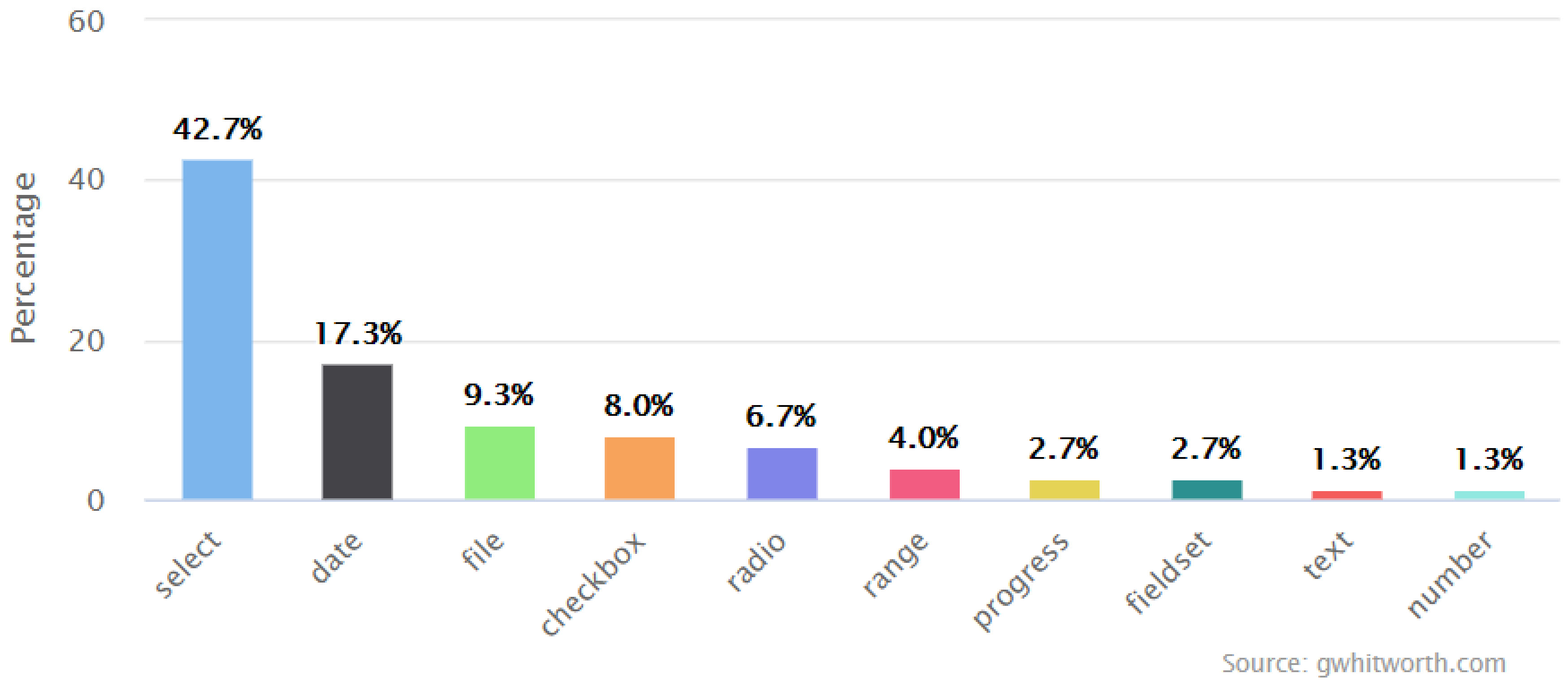
ionic

salesforce

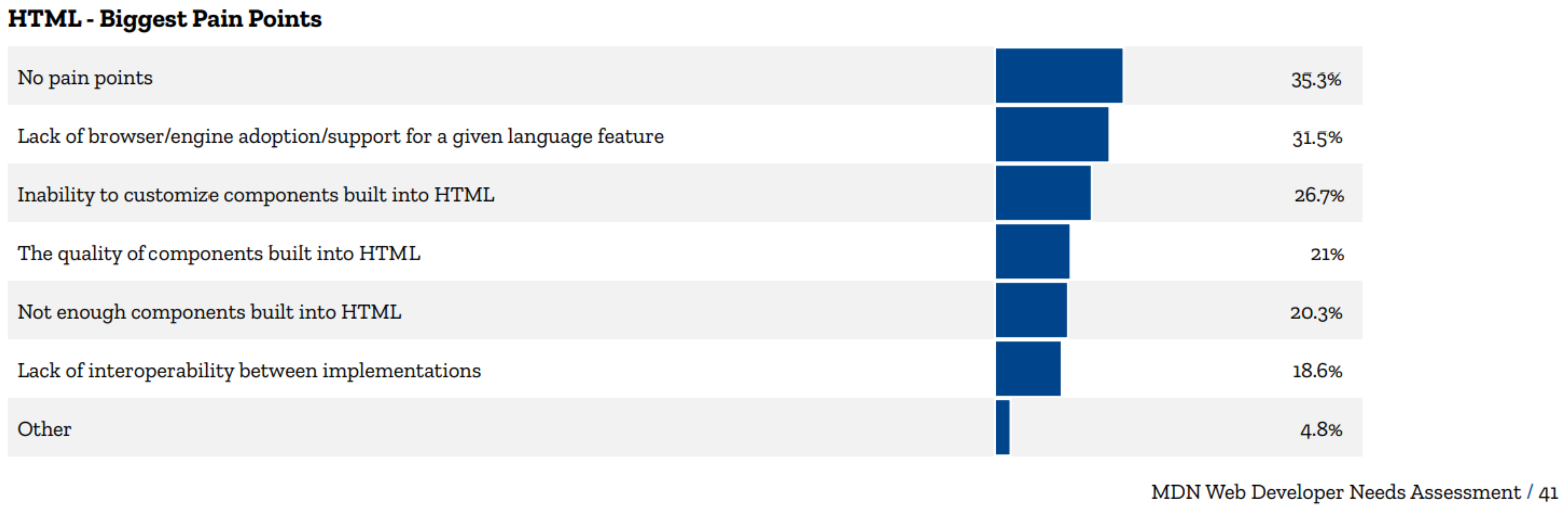
LinkedIn

Recreating controls causes developer pain

Survey results: most frustrating form controls



MDN Web DNA Survey 2019 results: HTML pain points

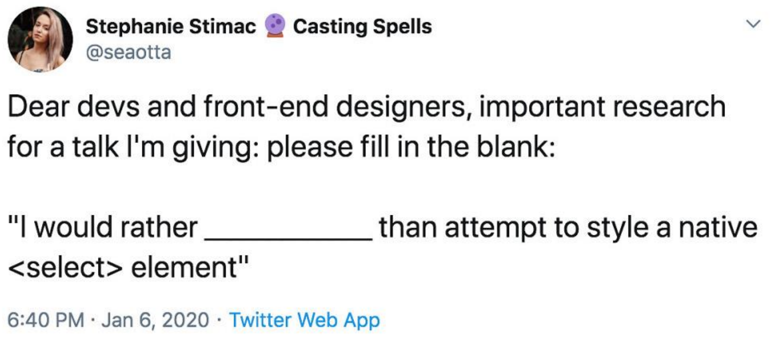


HTML - Biggest Pain Points

No pain points	35.3%
Lack of browser/engine adoption/support for a given language feature	31.5%
Inability to customize components built into HTML	26.7%
The quality of components built into HTML	21%
Not enough components built into HTML	20.3%
Lack of interoperability between implementations	18.6%
Other	4.8%

MDN Web Developer Needs Assessment / 41

Twitter complaints about controls, part I

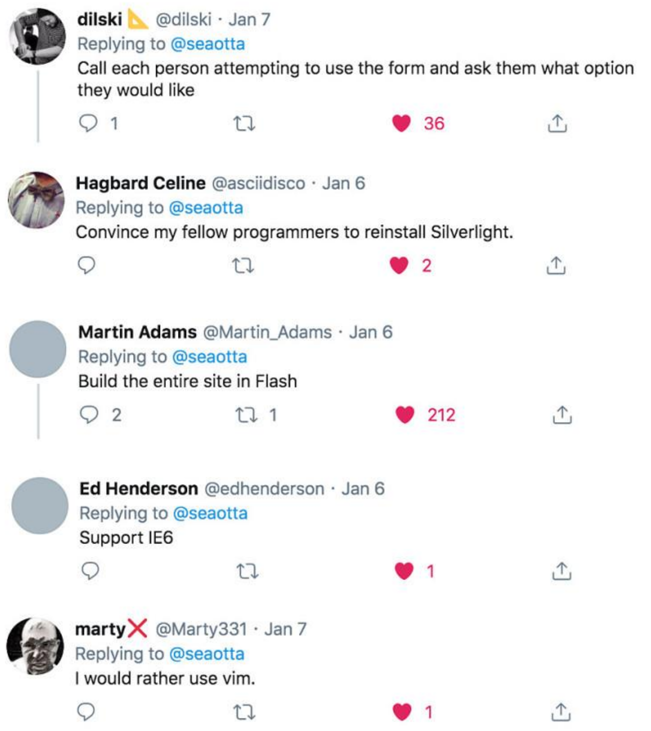


Stephanie Stimac 🔮 Casting Spells @seaotta

Dear devs and front-end designers, important research for a talk I'm giving: please fill in the blank:

"I would rather __________ than attempt to style a native <select> element"

6:40 PM · Jan 6, 2020 · Twitter Web App



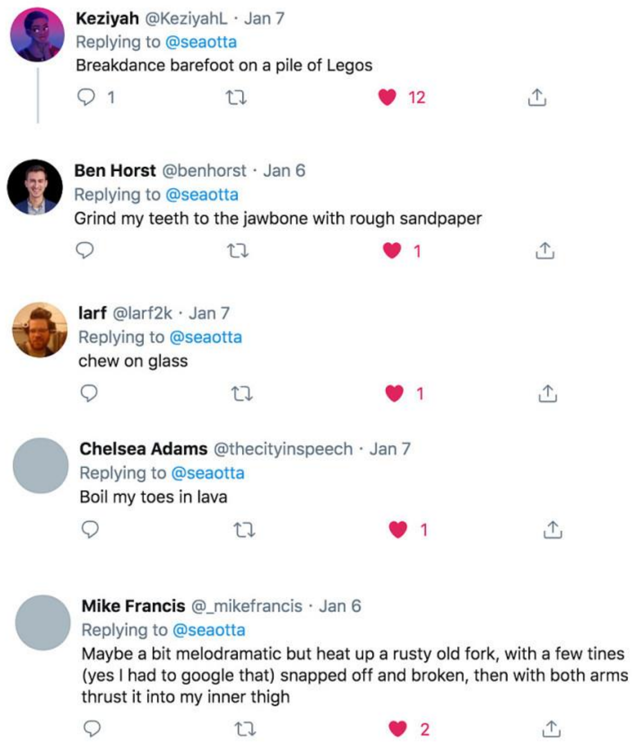
dilski 📐 @dilski · Jan 7
Replying to @seaotta
Call each person attempting to use the form and ask them what option they would like
1 · 36

Hagbard Celine @asciidisco · Jan 6
Replying to @seaotta
Convince my fellow programmers to reinstall Silverlight.
2

Martin Adams @Martin_Adams · Jan 6
Replying to @seaotta
Build the entire site in Flash
2 · 1 · 212

Ed Henderson @edhenderson · Jan 6
Replying to @seaotta
Support IE6
1

marty❌ @Marty331 · Jan 7
Replying to @seaotta
I would rather use vim.
1



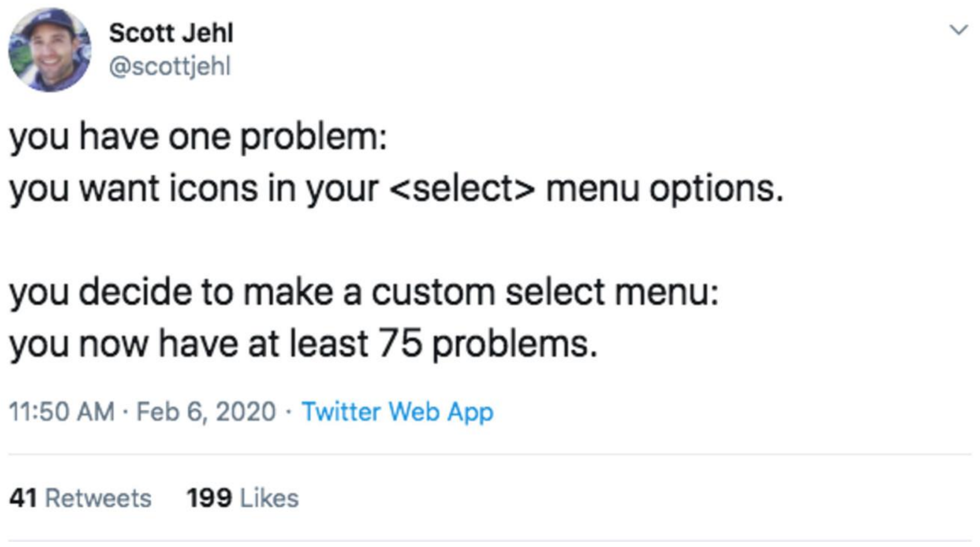
Keziyah @KeziyahL · Jan 7
Replying to @seaotta
Breakdance barefoot on a pile of Legos
1 · 12

Ben Horst @benhorst · Jan 6
Replying to @seaotta
Grind my teeth to the jawbone with rough sandpaper
1

larf @larf2k · Jan 7
Replying to @seaotta
chew on glass
1

Chelsea Adams @thecityinspeech · Jan 7
Replying to @seaotta
Boil my toes in lava
1

Mike Francis @_mikefrancis · Jan 6
Replying to @seaotta
Maybe a bit melodramatic but heat up a rusty old fork, with a few tines (yes I had to google that) snapped off and broken, then with both arms thrust it into my inner thigh
2



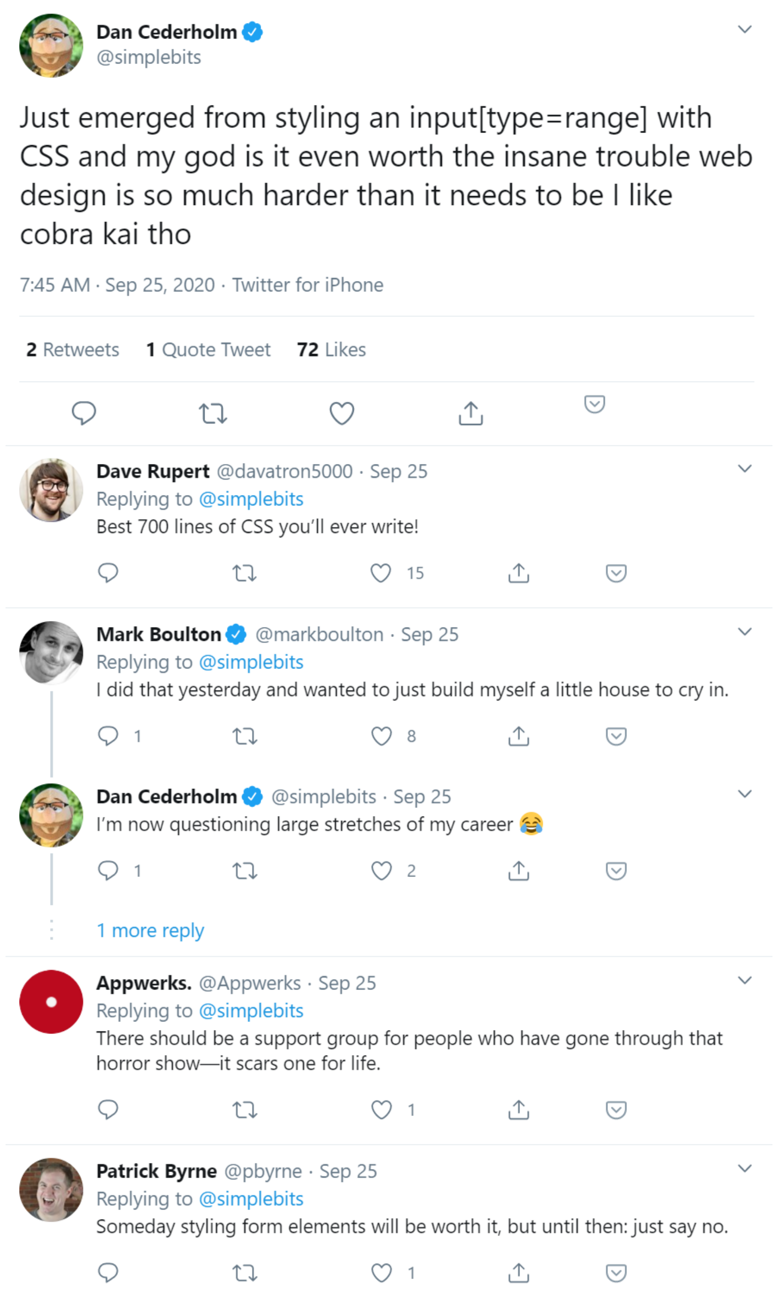
Scott Jehl
@scottjehl

you have one problem:
you want icons in your <select> menu options.

you decide to make a custom select menu:
you now have at least 75 problems.

11:50 AM · Feb 6, 2020 · Twitter Web App

41 Retweets 199 Likes



Dan Cederholm
@simplebits

Just emerged from styling an input[type=range] with CSS and my god is it even worth the insane trouble web design is so much harder than it needs to be I like cobra kai tho

7:45 AM · Sep 25, 2020 · Twitter for iPhone

2 Retweets **1** Quote Tweet **72** Likes

Dave Rupert @davatron5000 · Sep 25
Replying to @simplebits
Best 700 lines of CSS you'll ever write!
15

Mark Boulton @markboulton · Sep 25
Replying to @simplebits
I did that yesterday and wanted to just build myself a little house to cry in.
1 8

Dan Cederholm @simplebits · Sep 25
I'm now questioning large stretches of my career 😂
1 2

1 more reply

Appwerks. @Appwerks · Sep 25
Replying to @simplebits
There should be a support group for people who have gone through that horror show—it scars one for life.
1

Patrick Byrne @pbyrne · Sep 25
Replying to @simplebits
Someday styling form elements will be worth it, but until then: just say no.
1

Form Controls Opportunity Analysis: Participants

Research Recruiting

- Designer: 31
- Fullstack: 29
- Front-end: 14

Twitter

- Designer: 44
- Front-end: 112
- Back-end: 24
- Full-stack: 73

Form Controls Opportunity Analysis: Sample Questions

How satisfied are you with the experience of fully styling the <select>* popup window on desktop?

How important is it for you to be able to fully style the <select> popup window on desktop?

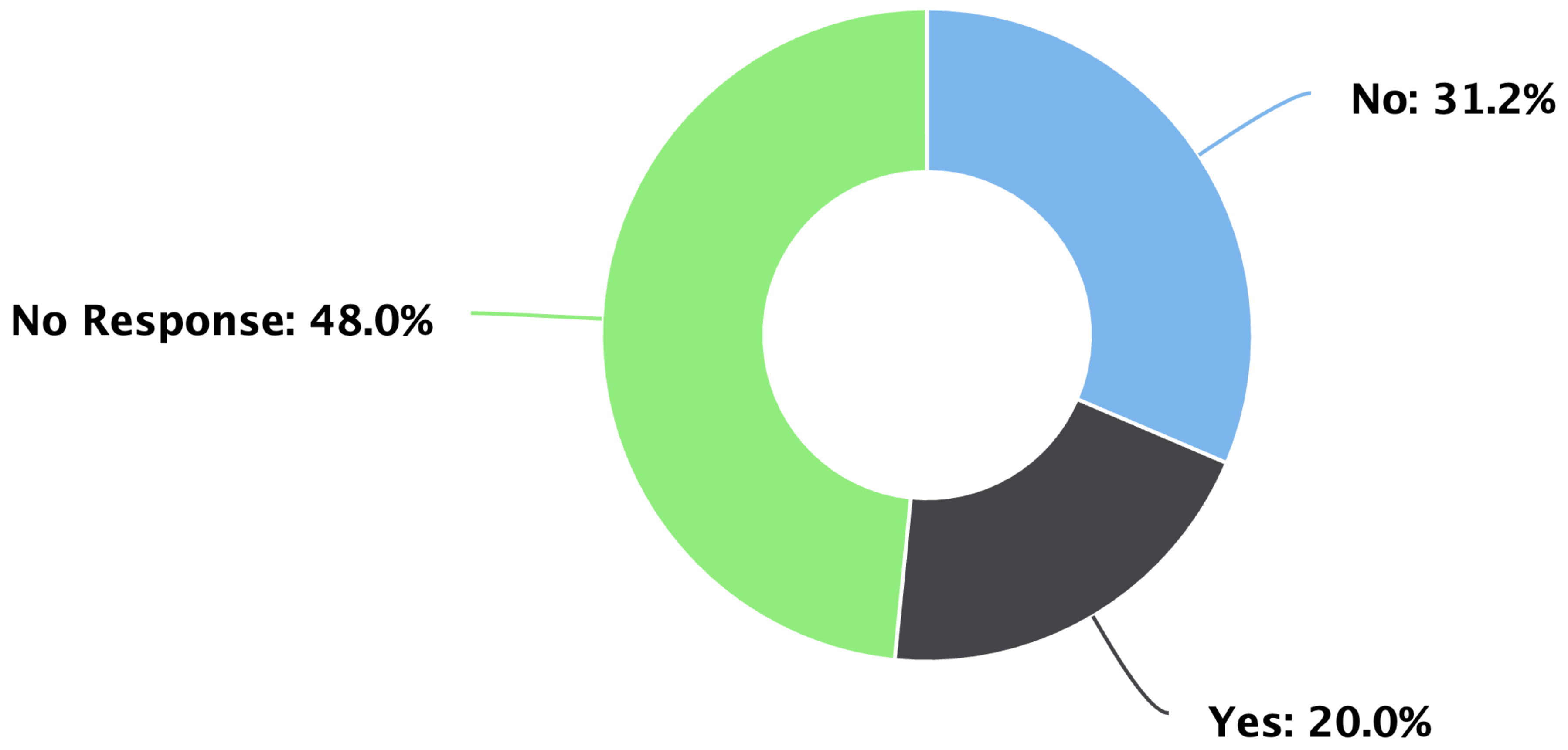
* This is just one example of the questions we asked regarding form control styling in our Opportunity Analysis. We inquired generically and about other form factors.

Form Controls Opportunity Analysis

Appearance	Frequency (building your own)	Effort (building your own)	Interop
10.36	10.48	13.54	15.23

Recreating controls causes user experience issues

Survey results: many web developers are not testing for accessibility

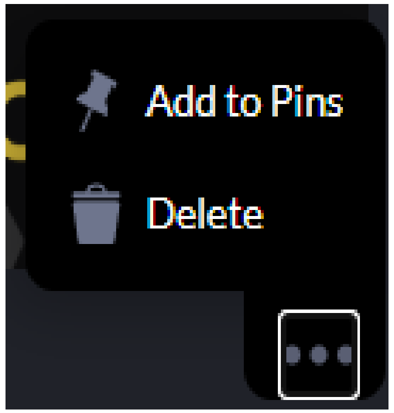


Source: gwhitworth.com

A sampling of accessibility issues in the wild

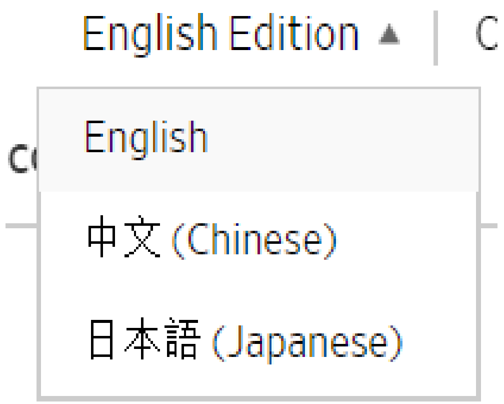
Limited keyboard support

Alexa Rank: 1,247



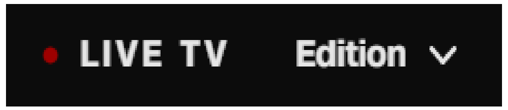
Limited keyboard or AT support

Rank: 454



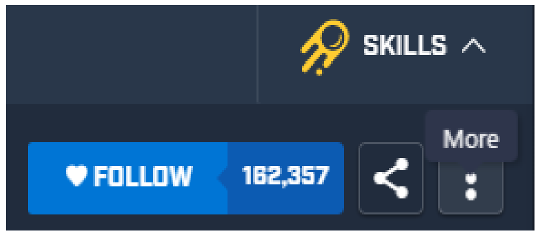
Not focusable via Keyboard

Alexa Rank: 96

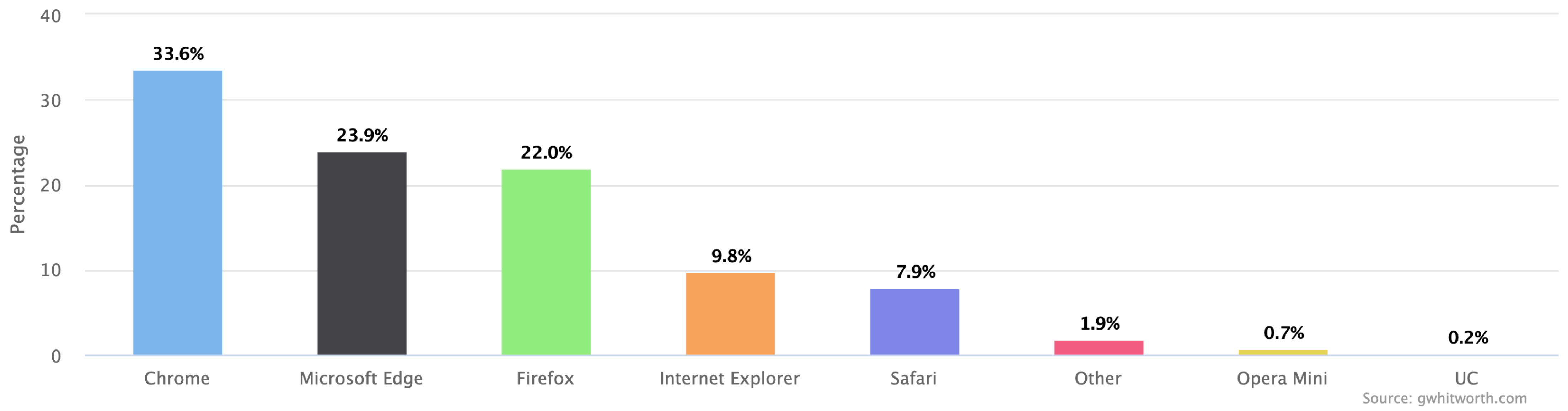


Not focusable via Keyboard nor AT Support

Alexa Rank: 1,293

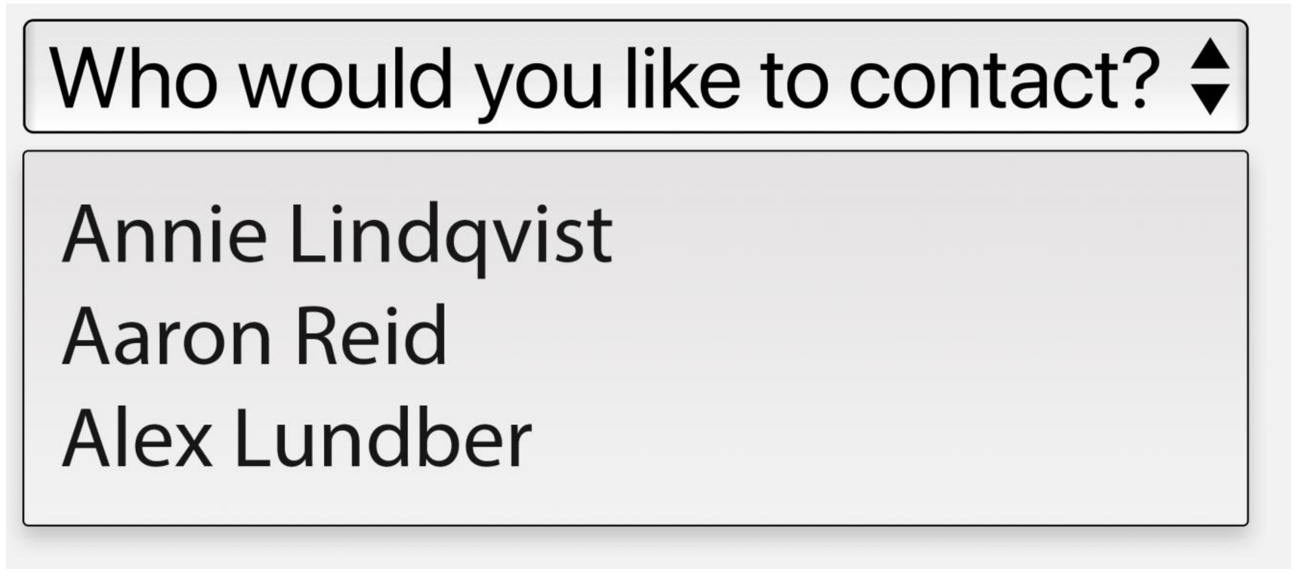


Survey results: browsers tested during development



What is a Control?

What *is* a form control, anyway?



An MVC model for form controls

Model

Data members and capabilities of the control, available to script. Examples:

- `value`
- Form association
- Validity state

View

The user interface. Exposes state to the user, enables the user to interact with the control and change state.

Controller

Intermediary between model and view.

- Inform model of interactions with view, via input handlers
- Inform view of changes to model, via events, CSS pseudo selectors, exposed properties

Our current mental model of a control

A control is a type of component that manages user interaction.

The control has controller code that manages changes in the component's states and its model based on user interaction with its parts.

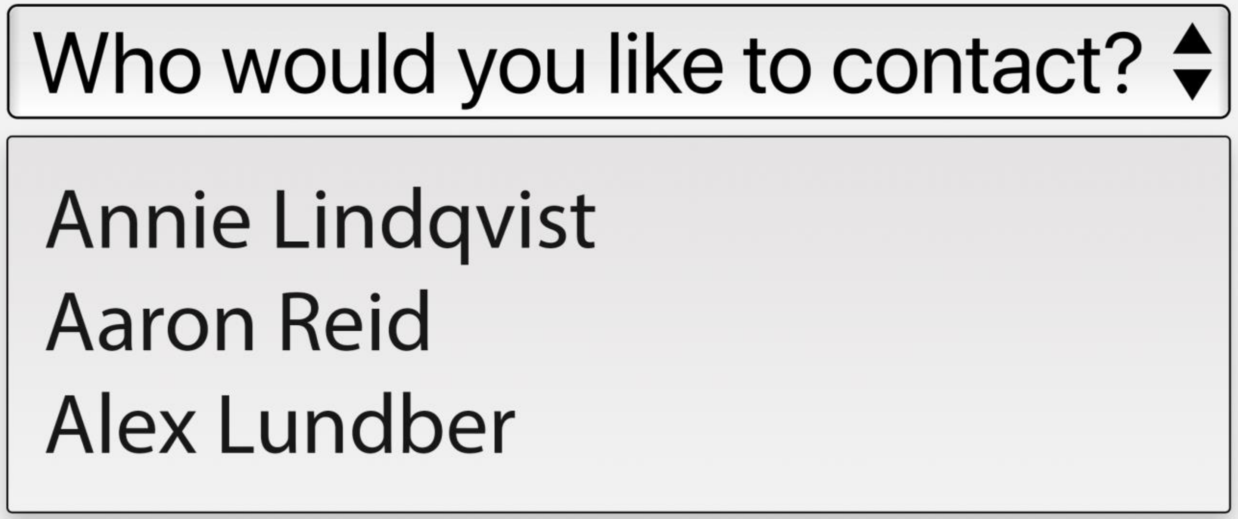
Spectrum of Customization

Controls currently fall along a customization spectrum

Less Customizability ⟷ More

None	Hint	Limited	Fully Style-able (In some cases, minor limitations)	Fully Extensible
color picker	accent-color	range	text inputs	
date picker	color-scheme	file inputs	button	
datalist		meter	output	
file picker		option	Label	
select picker		optgroup	summary	
		Progress		
		checkboxes*		
		radios*		
		select		

Level of customization: **none**



Who would you like to contact?

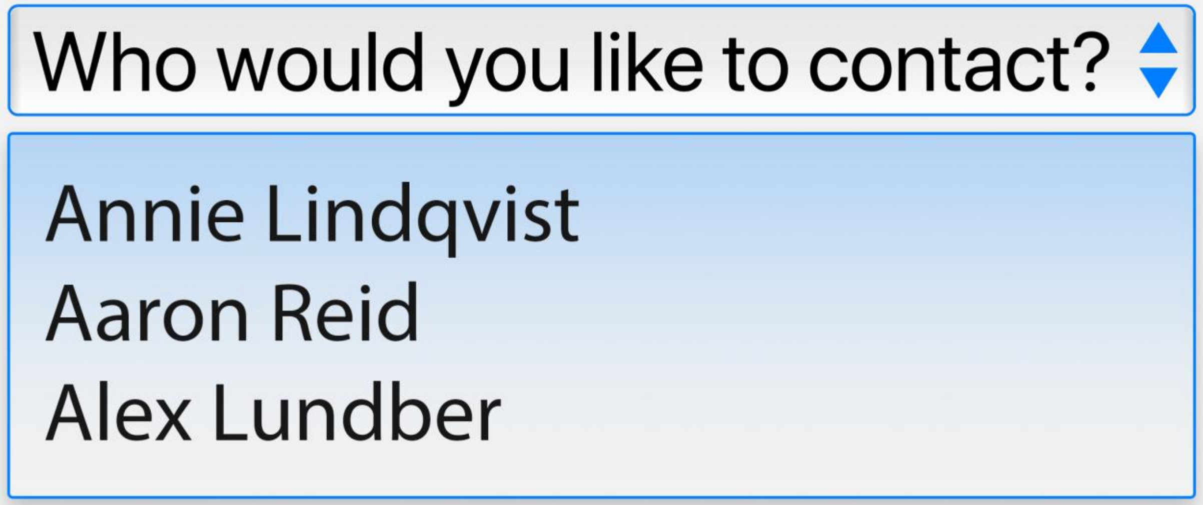
Annie Lindqvist

Aaron Reid

Alex Lundber

Interop needed: none

Level of customization: **hint**

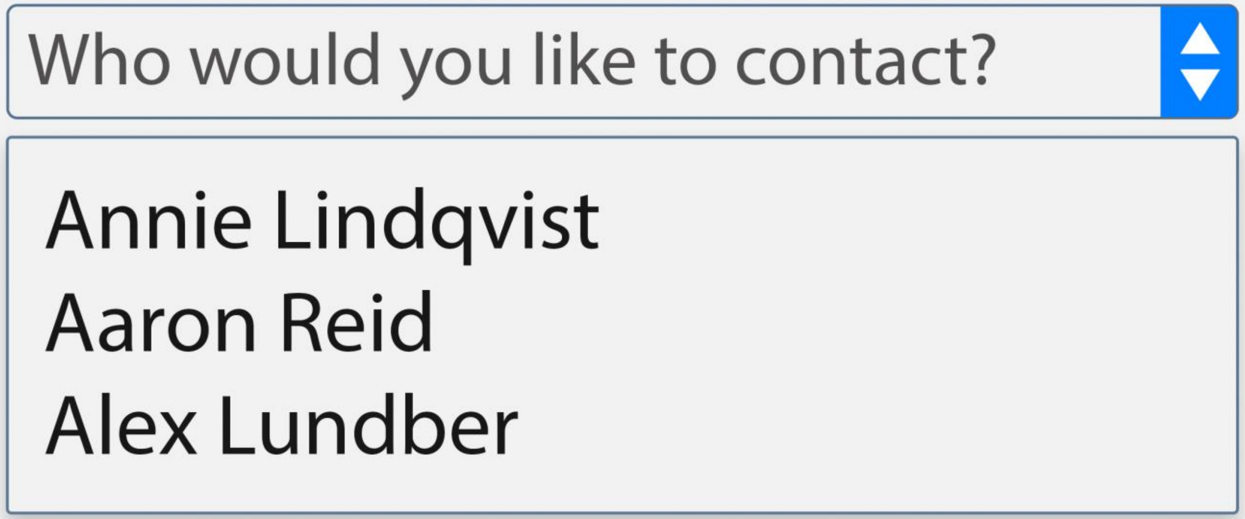

Who would you like to contact?

Annie Lindqvist
Aaron Reid
Alex Lundber

```
select {
  accent-color: #007eff;
}
```

Interop needed: none

Level of customization: **limited**

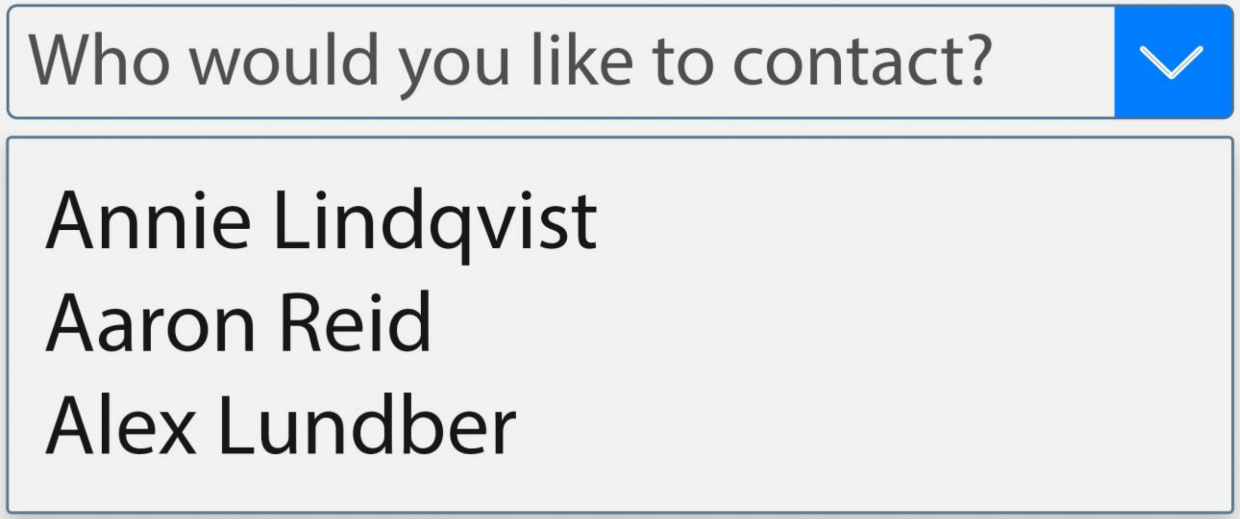

Who would you like to contact?

Annie Lindqvist
Aaron Reid
Alex Lundber

```
select::select-button {
  fill: white;
  background: #007eff
}
```

Interop needed: Limited

Level of customization: **limited**

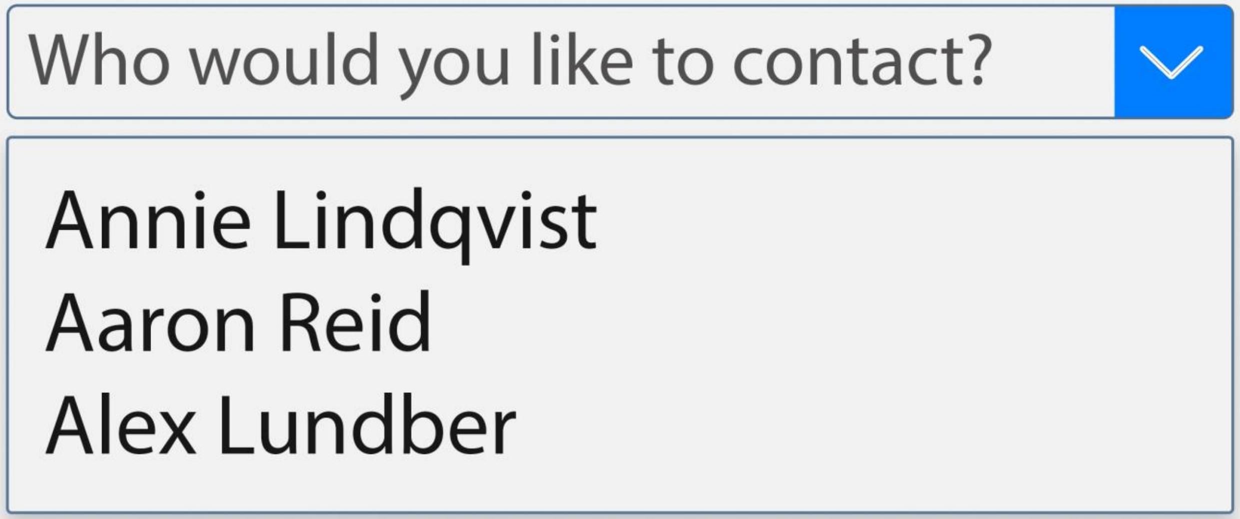

Who would you like to contact?

Annie Lindqvist
Aaron Reid
Alex Lundber

```
select::select-button {
  background-image: my-arrow.png;
  background-color: #007eff
}
```

Interop needed: Limited

Level of customization: **limited**

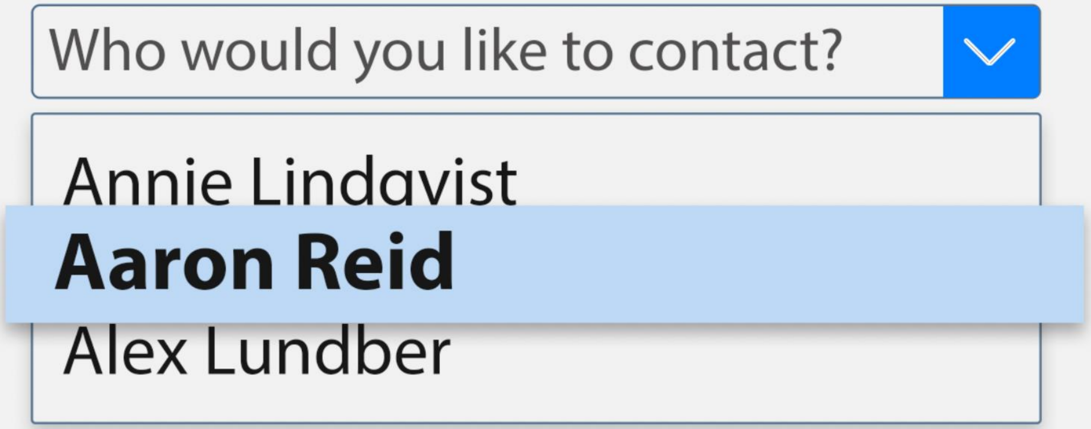


Why is this not possible today with the built-in <select>?

- Parts are not standardized
- Specific definition of what CSS properties are valid on the various parts

Interop needed: Limited

Level of customization: **Fully Style-able**

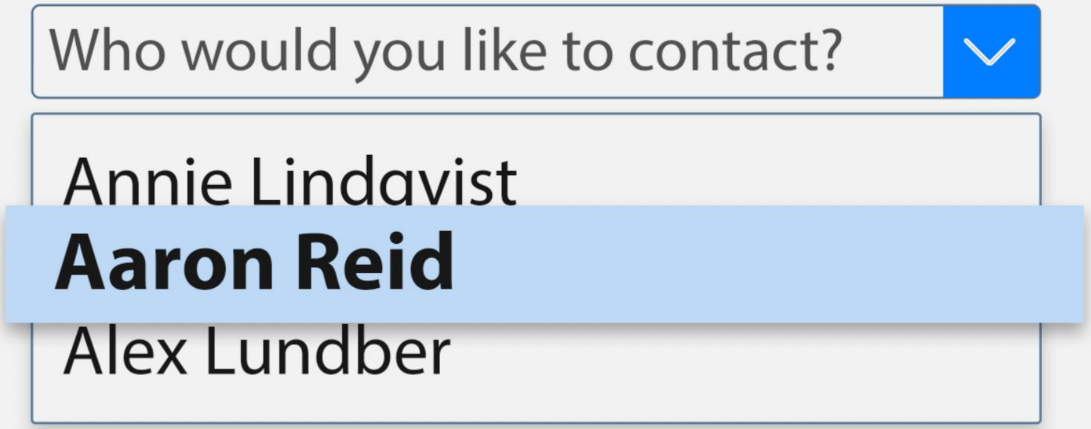


Why is this not possible today with the built-in <select>?

- Listbox is not standardized
- Listbox does not allow overflow
- Option elements don’t allow layout changes

Interop needed: High

Level of customization: **Fully Style-able**

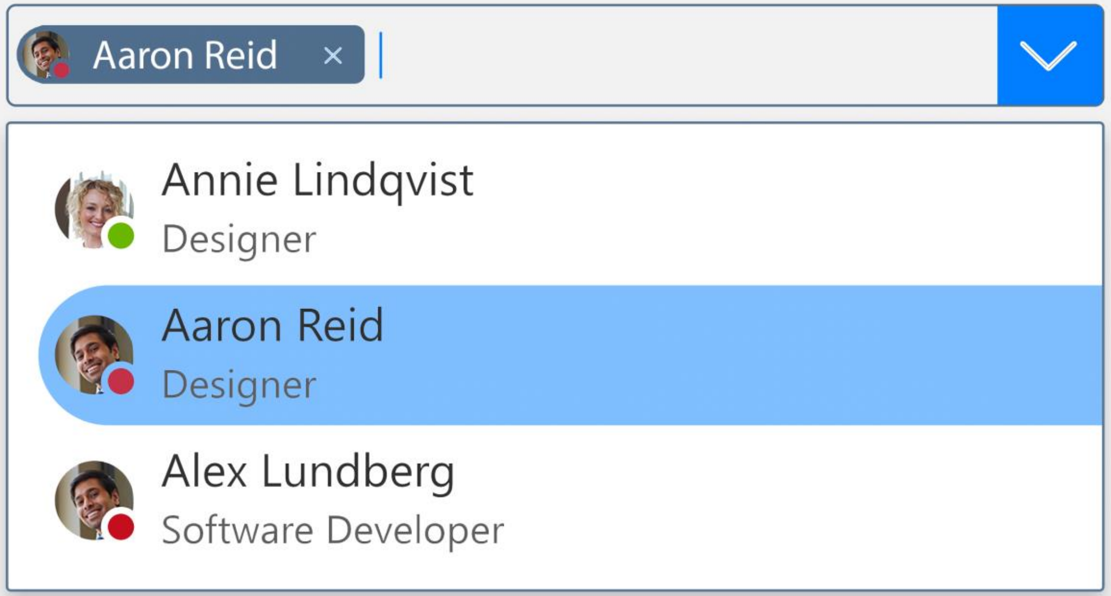


What is needed to enable this?

- Standardized parts
- Standardized DOM Structure
- Standardized Base styles

Interop needed: High

Level of customization: **Fully Extensible**



Why is this not possible today with the built-in <select>?

- Behaviors
- States
- Parts
- DOM Structure
- Structural styles

Interop needed: Full

Spectrum of Customizability

Less Customizability ⟷ More

None	Hint	Limited	Fully Style-able	Fully Extensible
Does not allow any style-ability or extensibility to the control or some of its parts	Allows the author to provide a value that the UA applies to a component or control that aligns with the spirit of the property	Pseudo elements, HTML elements, attributes that provide customizability but are limited in some manner	Elements that allow developers to opt-in to standardized parts, DOM structure, and base styles that user-agents apply their styles upon	Standardization of a control's anatomy, states, behaviors, with the capability of reusing controller code via defined parts

Solving fully style-able controls

An MVC model for form controls

Enable authors to use the platform

Model

Data members and capabilities of the control, available to script. Examples:

- value
- Form association
- Validity state

Enable authors to use the platform

Controller

Intermediary between model and view.

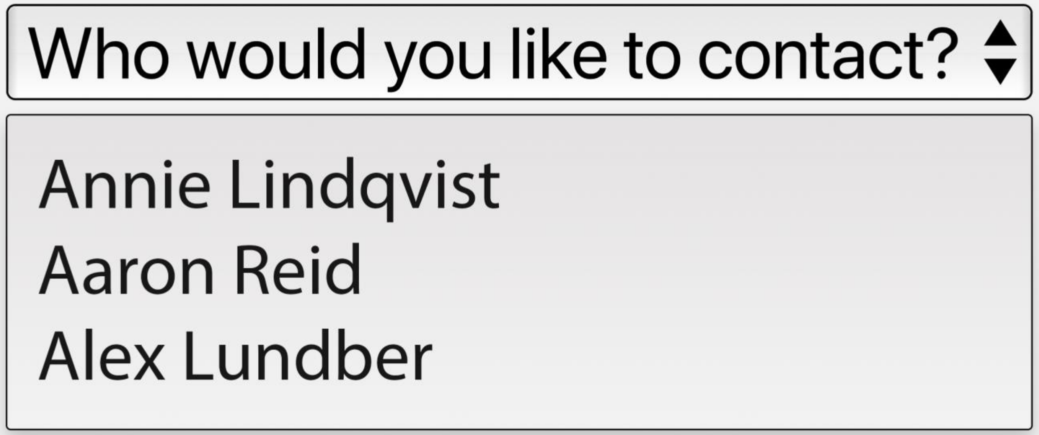
- Inform model of interactions with view, via input handlers
- Inform view of changes to model, via events, CSS pseudo selectors, exposed properties

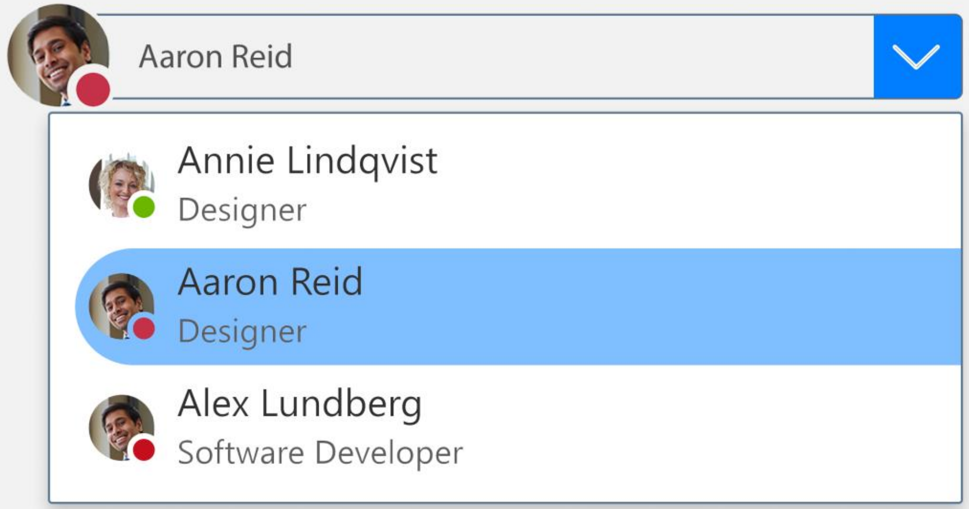
Enable authors to style

View

The user interface. Exposes state to the user, enables the user to interact with the control and change state.

How to enable standardized DOM & Styles





The custom attribute

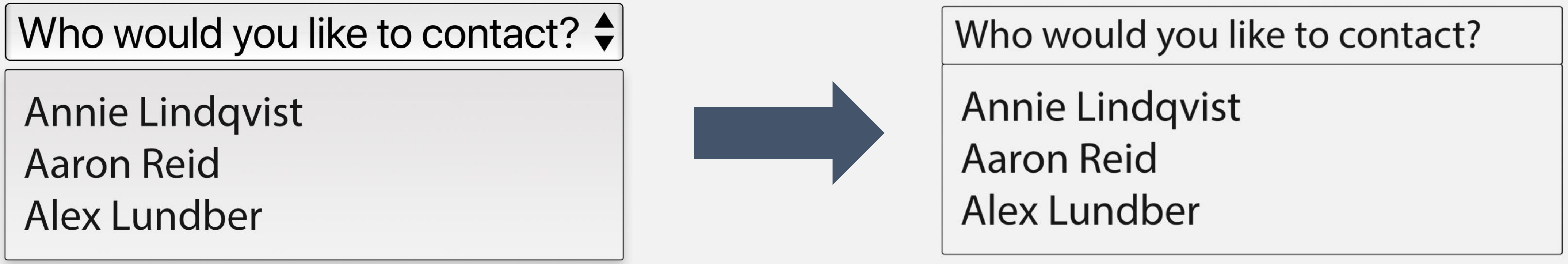
```
<select>
   <option></option>
</select>
```

The custom attribute modifies the DOM structure to be only the standardized parts, structural styles, and state styles.

The custom attribute

```
<select custom>
  <div part=“button-container”>
    <div part=“selected-value”></div>
  </div>
  <div part=“listbox-container”>
      <option></option>
  </div>
</select>
```

The custom attribute enables the structural stylesheet


Who would you like to contact?

Annie Lindqvist
Aaron Reid
Alex Lundber

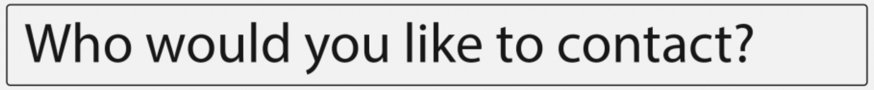
Who would you like to contact?

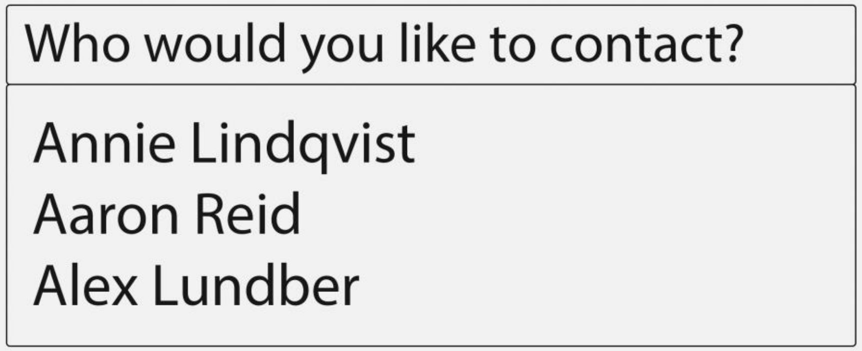
Annie Lindqvist
Aaron Reid
Alex Lundber

The custom attribute enables the structural stylesheet

```
select[custom] { }
```

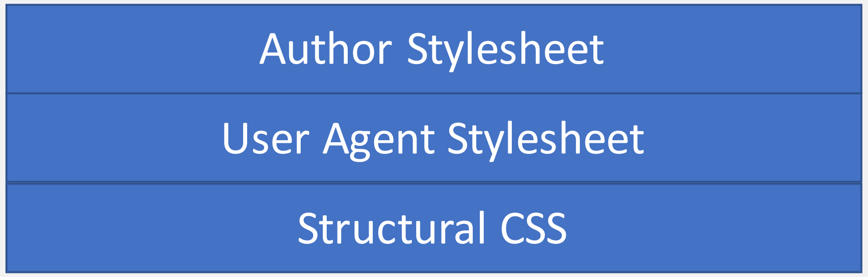
```
select[custom][open] { }
```


Who would you like to contact?


Who would you like to contact?

Annie Lindqvist
Aaron Reid
Alex Lundber

How to enable standardized DOM & Styles



“UAs should include in their user agent stylesheet style rules to give widgets a recognizable shape when appearance is none.”

And then there’s all: unset ☺ as well

Solving fully extensible controls

An MVC model for form controls

Enable authors to use the platform

Model

Data members and capabilities of the control, available to script. Examples:

- value
- Form association
- Validity state

Enable authors to use the platform

Controller

Intermediary between model and view.

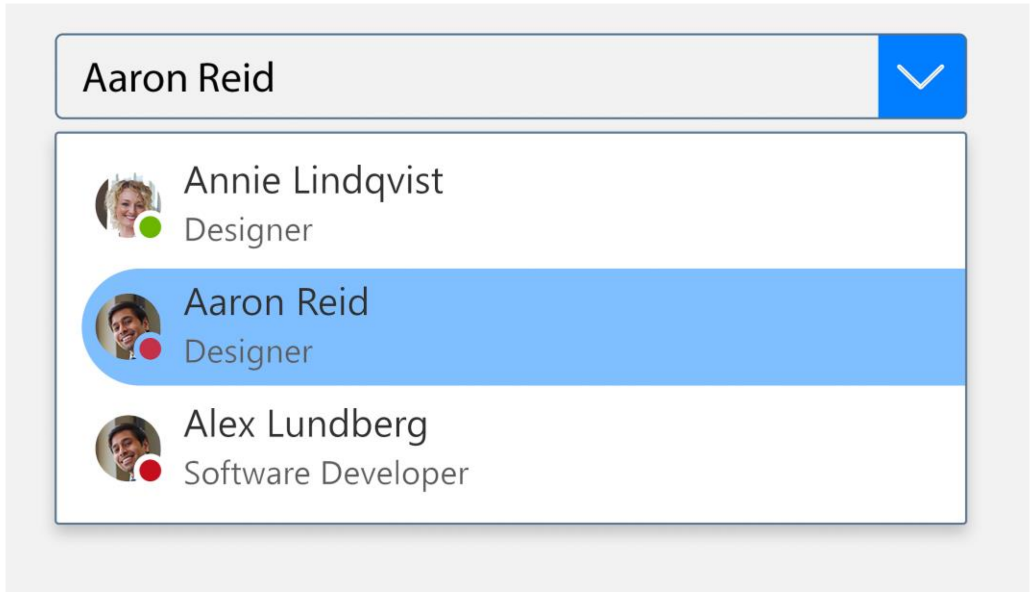
- Inform model of interactions with view, via input handlers
- Inform view of changes to model, via events, CSS pseudo selectors, exposed properties

Enable authors to extend

View

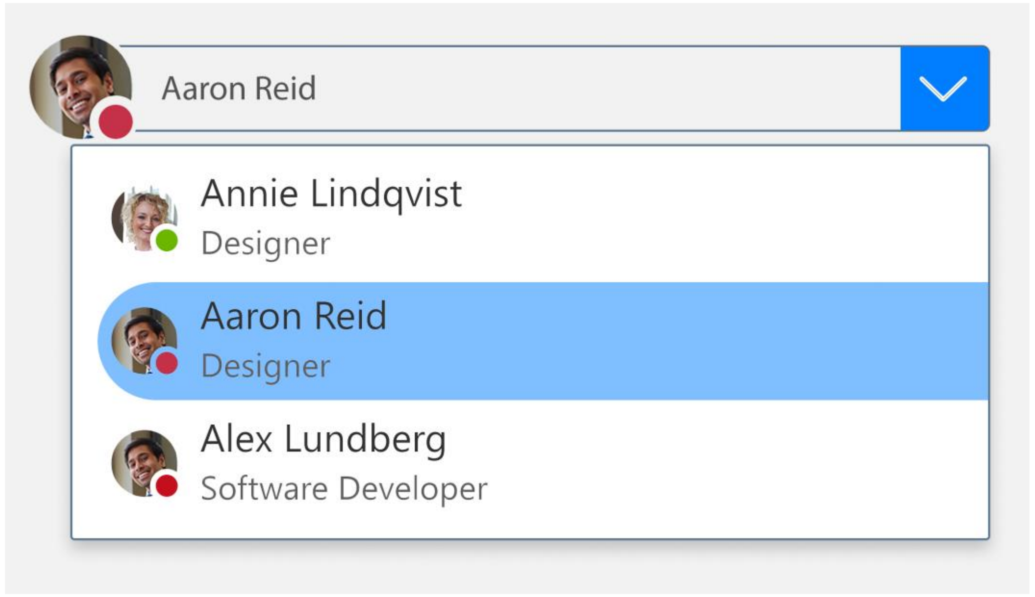
The user interface. Exposes state to the user, enables the user to interact with the control and change state.

Web devs can update `slots` to replace the content of control parts...



```
<div slot="listbox" part="listbox" class="my-box">
    <option>
      <img src="annie.jpg" alt="" />
      <p class="name">Annie Lindqvist</p>
      <p class="title">Designer</p>
      <span class="status">Online</span>
    </option>
    ...
</div>
```

...or they can replace the entire view with their own Shadow DOM



```
let customSelect =
document.createElement('select');

customSelect.setAttribute("custom", "");

let selectShadow = customSelect.attachShadow({
mode: 'open' });

selectShadow.innerHTML = `My custom select UI`;
document.body.appendChild(customSelect);
```

Core parts of the author's shadow DOM **must** be labeled with the `part` attribute:
`part="button"`, `part="listbox"`

Parts help the web dev leverage the platform

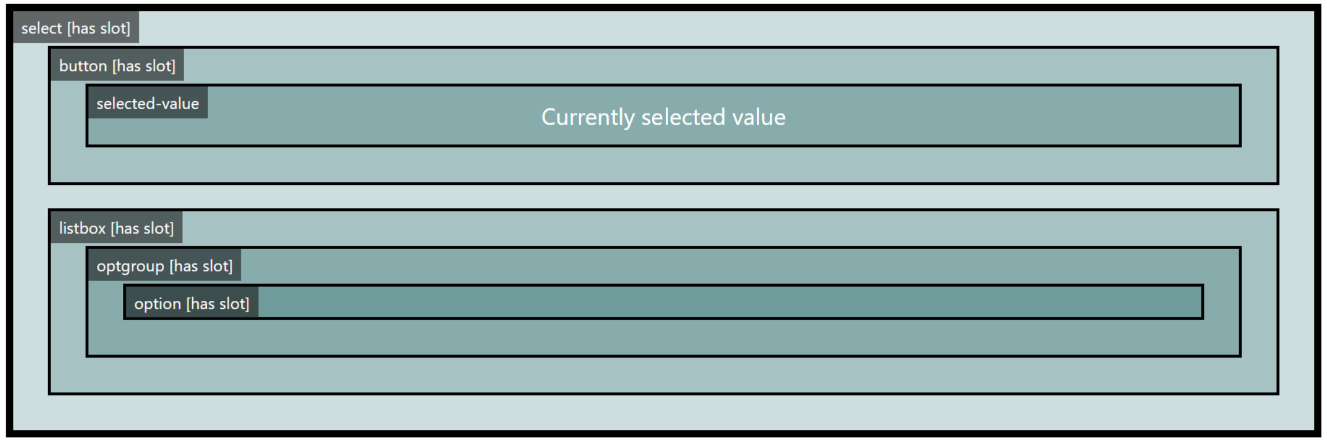
Controller code can:

1. Inform the model of user interactions with the view, e.g. selected `option`

2. Apply the right intrinsic accessibility semantics, e.g. part-appropriate roles, states, and properties

3. Wire up the correct behaviors, e.g. keyboard interactions for opening and closing the select popup, traversing through options, etc.

Extensibility requires standardized DOM structure



Structure

- `<select>` - The root element that contains the button and listbox **[required]**
- `<button>` - The button element that contains the selected value and triggers the visibility of the listbox **[required]**
- `<listbox>` - The wrapper that contains the `<option>`(s) and `<optgroup>`(s) **[required]**
- `<optgroup>` - Groups `<options>` together with a label **[optional]**
- `<option>` - Can have one or more and represents the potential values that can be chosen by the user **[required]**

Content not allowed within the anatomy

The following interactive elements are **NOT** permitted within a `<select>` or its children:

- button (outside of the pre-defined one)
- datalist
- input (all types)
- meter
- progress
- select

Extensibility requires standardized states

States

open

This state is applied to the `<select>` when the listbox is visible to the user.

required

An `<option>` from the `<select>` must be selected when the `required` attribute is set to `true`

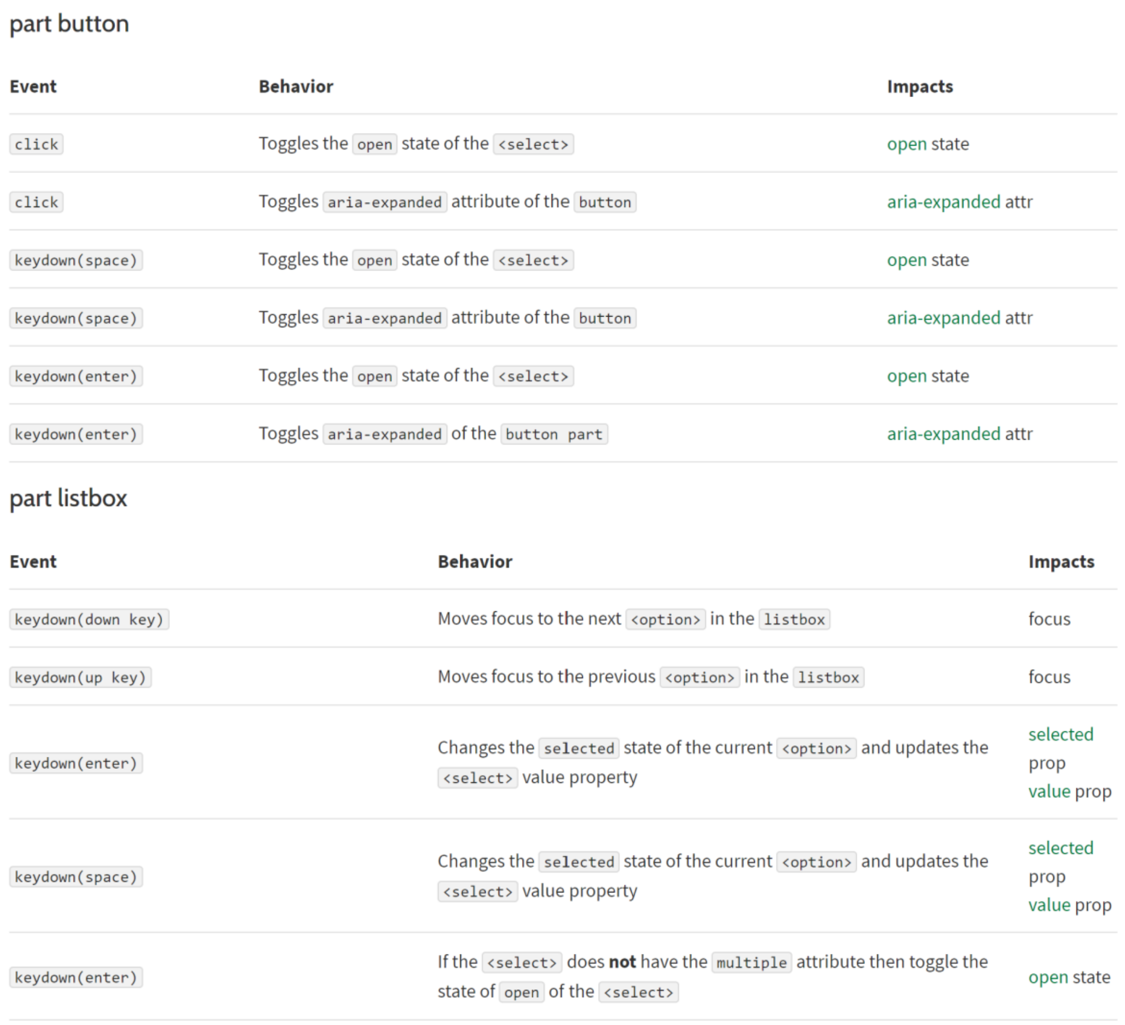
valid

The `<select>` meets all its validation constraints, and is therefore considered to be valid.

invalid

The `<select>` does not meet its validation constraints, and is therefore considered to be invalid.

Extensibility requires standardized behaviors

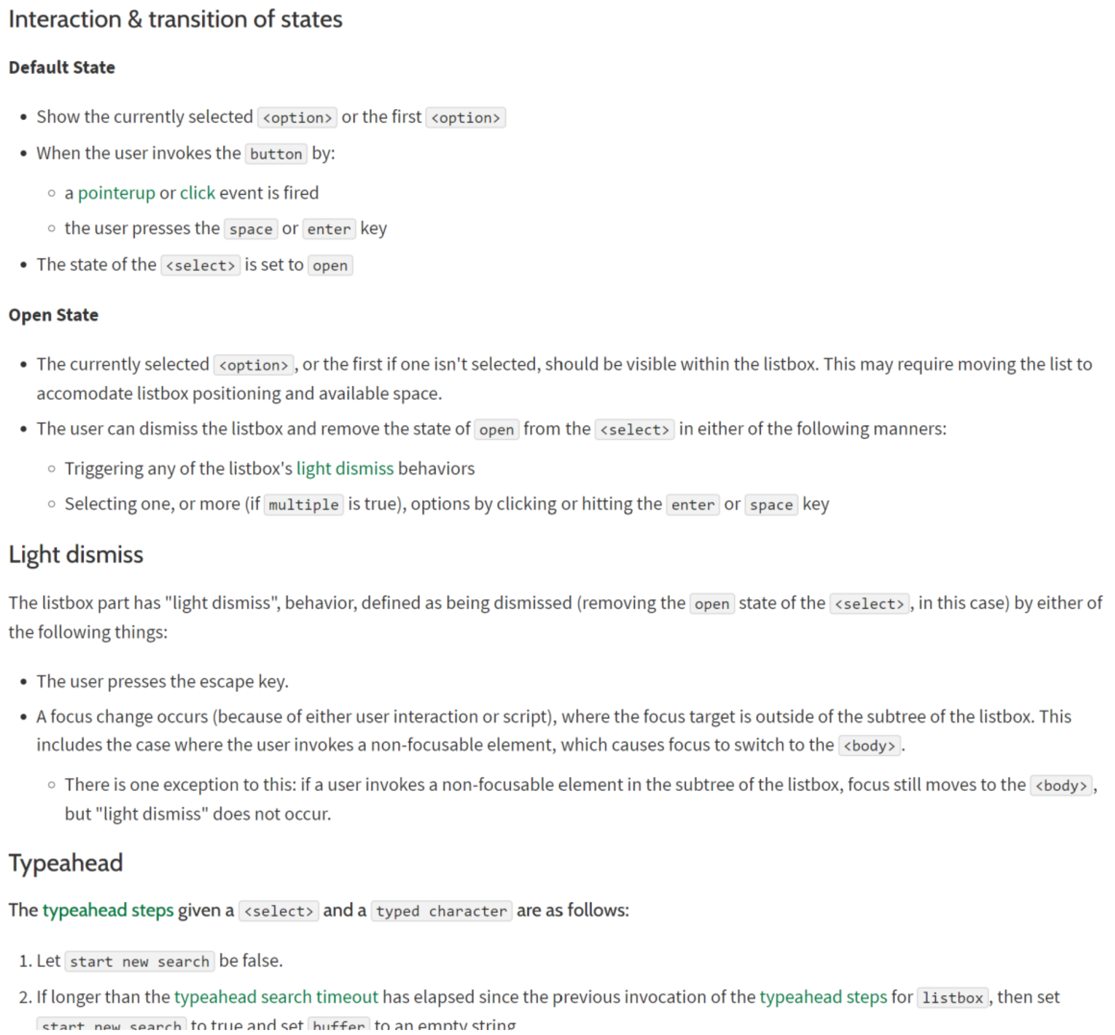


part button

Event	Behavior	Impacts
`click`	Toggles the `open` state of the `<select>`	open state
`click`	Toggles `aria-expanded` attribute of the `button`	aria-expanded attr
`keydown(space)`	Toggles the `open` state of the `<select>`	open state
`keydown(space)`	Toggles `aria-expanded` attribute of the `button`	aria-expanded attr
`keydown(enter)`	Toggles the `open` state of the `<select>`	open state
`keydown(enter)`	Toggles `aria-expanded` of the `button part`	aria-expanded attr

part listbox

Event	Behavior	Impacts
`keydown(down key)`	Moves focus to the next `<option>` in the `listbox`	focus
`keydown(up key)`	Moves focus to the previous `<option>` in the `listbox`	focus
`keydown(enter)`	Changes the `selected` state of the current `<option>` and updates the `<select>` value property	selected prop value prop
`keydown(space)`	Changes the `selected` state of the current `<option>` and updates the `<select>` value property	selected prop value prop
`keydown(enter)`	If the `<select>` does **not** have the `multiple` attribute then toggle the state of `open` of the `<select>`	open state



Interaction & transition of states

Default State

- Show the currently selected `<option>` or the first `<option>`
- When the user invokes the `button` by:
  - a pointerup or click event is fired
  - the user presses the `space` or `enter` key
- The state of the `<select>` is set to `open`

Open State

- The currently selected `<option>`, or the first if one isn't selected, should be visible within the listbox. This may require moving the list to accomodate listbox positioning and available space.
- The user can dismiss the listbox and remove the state of `open` from the `<select>` in either of the following manners:
  - Triggering any of the listbox's light dismiss behaviors
  - Selecting one, or more (if `multiple` is true), options by clicking or hitting the `enter` or `space` key

Light dismiss

The listbox part has "light dismiss", behavior, defined as being dismissed (removing the `open` state of the `<select>`, in this case) by either of the following things:

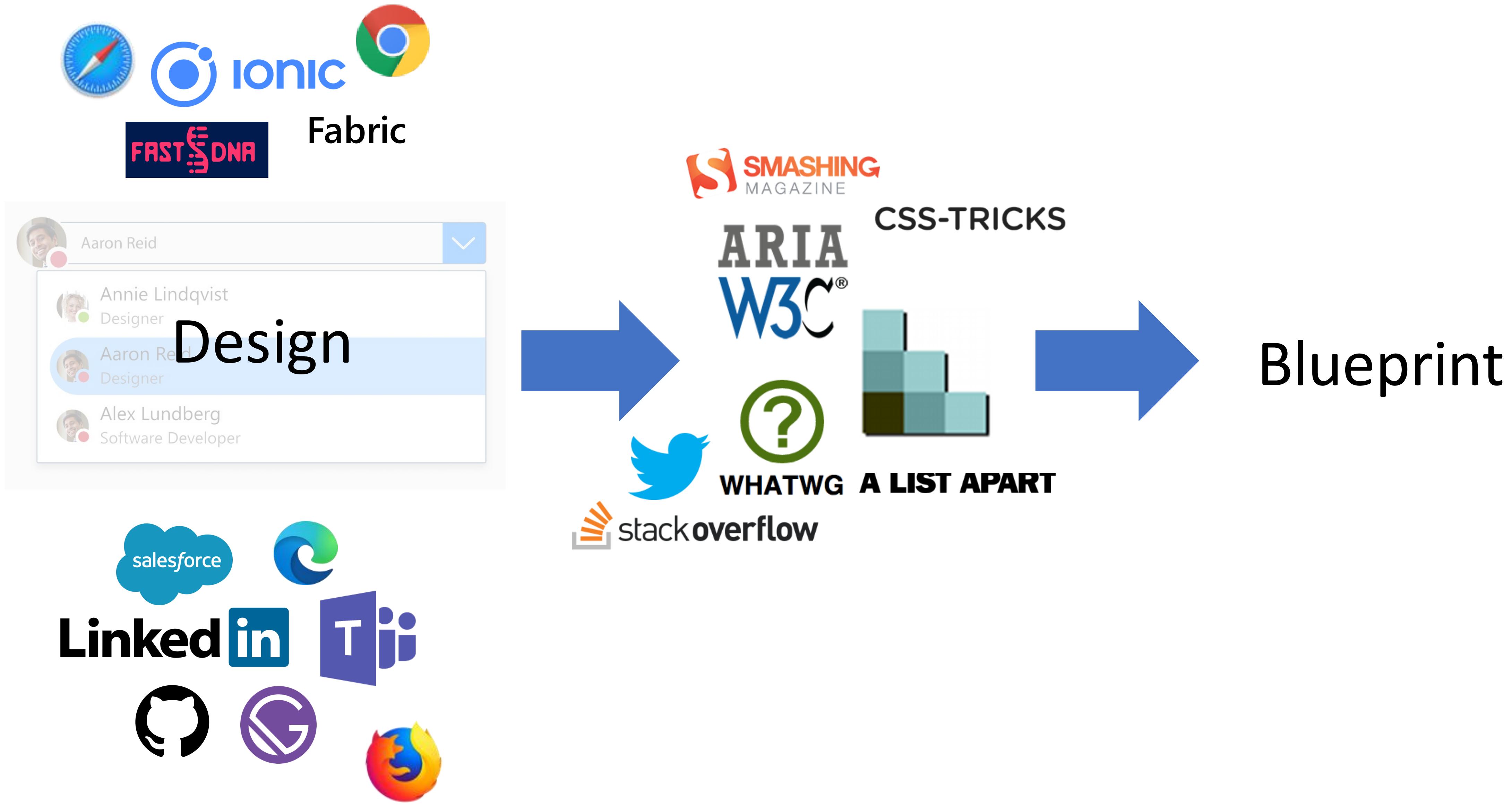
- The user presses the escape key.
- A focus change occurs (because of either user interaction or script), where the focus target is outside of the subtree of the listbox. This includes the case where the user invokes a non-focusable element, which causes focus to switch to the `<body>`.
  - There is one exception to this: if a user invokes a non-focusable element in the subtree of the listbox, focus still moves to the `<body>`, but "light dismiss" does not occur.

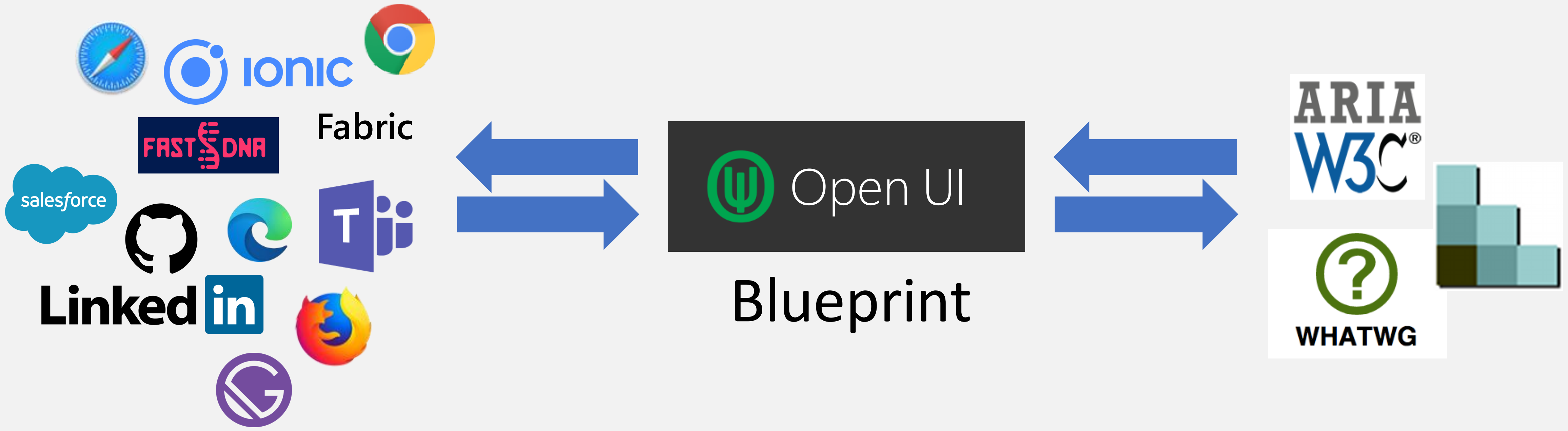
Typeahead

The typeahead steps given a `<select>` and a `typed character` are as follows:

1. Let `start new search` be false.
2. If longer than the typeahead search timeout has elapsed since the previous invocation of the typeahead steps for `listbox`, then set `start new search` to true and set `buffer` to an empty string

Process for holistic control standardization


ionic
FAST DNA
Fabric
Aaron Reid
Annie Lindqvist
Designer
Aaron Reid
Designer
Alex Lundberg
Software Developer
Design
salesforce
Linked in
SMASHING MAGAZINE
CSS-TRICKS
ARIA
W3C
WHATWG
A LIST APART
stackoverflow
Blueprint


ionic
Fabric
FAST DNA
salesforce
Linked in
T
Open UI
Blueprint
ARIA
W3C®
WHATWG

Discussion

Discussion Agenda

Break (15 min)

Problem (15 min)

Control Definition (15 min)

Spectrum (30 min)

Break (15 min)

Solutions for fully style-able/extensible, standardization (1 hour)

Process (30 min)

Discussion: The Problem

Proposed resolution: The Problem

Web developers recreating a browser's form controls is a problem.

Proposed resolution: The problem

Web standards should improve control customizability, so that developers don't need to recreate these from the ground up.

Discussion: Definition of a control

Proposed resolution: Control Definition

A control is a type of component that manages user interaction.

The control has controller code that manages changes in the component's states and its model based on user interaction with its parts.

Discussion: Spectrum of customization

Spectrum of Customizability

Less Customizability ⟷ More

None	Hint	Limited	Fully Style-able	Fully Extensible
Does not allow any style-ability or extensibility to the control or some of its parts	Allows the author to provide a value that the UA applies to a component or control that aligns with the spirit of the property	Pseudo elements, HTML elements, attributes that provide customizability but are limited in some manner	Elements that allow developers to opt-in to standardized anatomy, DOM structure, and base styles that user-agents apply their styles upon	Standardization of a control's anatomy, states, behaviors, with the capability of reusing controller code via defined parts

Proposed resolution: we will normatively define a spectrum of customizability for controls

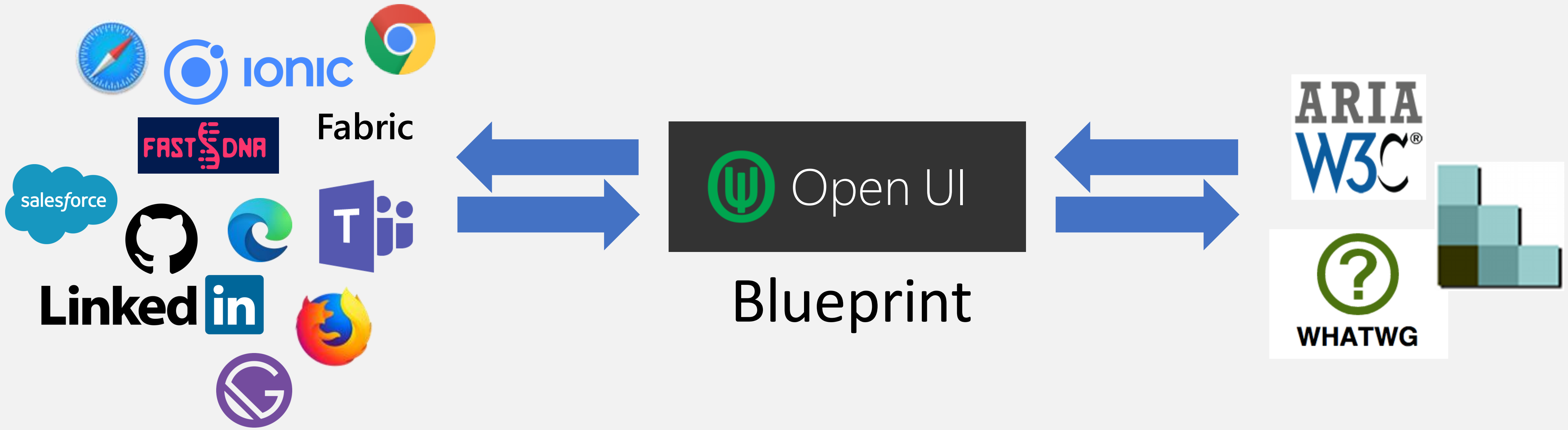
What parts do you feel are necessary to explore to solve this problem? What is necessary to avoid re-creating controls?

Less Customizability ⟷ More

None	Hint	Limited	Fully Style-able	Fully Extensible
Does not allow any style-ability or extensibility to the control or some of its parts	Allows the author to provide a value that the UA applies to a component or control that aligns with the spirit of the property	Pseudo elements, HTML elements, attributes that provide customizability but are limited in some manner	Elements that allow developers to opt-in to standardized parts, DOM structure, and base styles that user-agents apply their styles upon	Standardization of a control's anatomy, states, behaviors, with the capability of reusing controller code via defined parts
A	B	C	D	E

Proposed resolution

We will standardize control anatomy (`slots` and `parts`), states, and behaviors.


ionic
Fabric
FAST DNA
salesforce
Linked in
Open UI
Blueprint
ARIA
W3C®
WHATWG

Proposed resolution

Control definitions will begin in Open UI to have a complete specification created for all parts, states and behaviors.

New CSS pseudo elements, classes or primitives will be standardized in the CSSWG. New elements, attributes or DOM events will be standardized in WHATWG. New ARIA roles will be standardized in the ARIA WG.

Appendix

Resources

- Initial thoughts on standardizing form controls
- Can we please style the <select> control?!
- Customizing control UI explainer

- Recording of the presentation by Melanie Richards and Greg Whitworth

Overall Needs Ranking

One is the most frustrating and 28 is the least frustrating.

1. Having to support specific browsers (e.g., IE11).
2. Outdated or inaccurate documentation for frameworks and libraries.
3. Avoiding or removing a feature that doesn't work across browsers.
4. Testing across browsers.
5. Making a design look/work the same across browsers.
6. Discovering bugs not caught during testing.
7. Supporting multiple frameworks in the same code base.
8. Keeping up with a large number of new and existing tools or frameworks.
9. Managing user data to comply with laws and regulations.
10. Understanding and implementing security measures.
11. Integrating with third parties for authentication.
12. Pinpointing existing performance issues.
13. Running end-to-end tests.
14. Lack of device APIs allowing for access to hardware.
15. Outdated documentation for HTML, CSS and JavaScript.
16. Determining the root cause of a bug.
17. Capability of the web to support a specified layout.
18. Knowing what browsers support a specific technology.
19. Achieving visual precision on stylized elements (e.g., buttons).
20. Running front-end tests.
21. Implementing localization.
22. Keeping up with changes to the web platform.
23. Implementing performance optimizations.
24. Making sites accessible.
25. Getting users to grant permissions to Web APIs (e.g., geo-location).
26. Deciding what to learn next to keep my skill set relevant.
27. Finding a community of peers.
28. Fixing a bug once it's been identified.