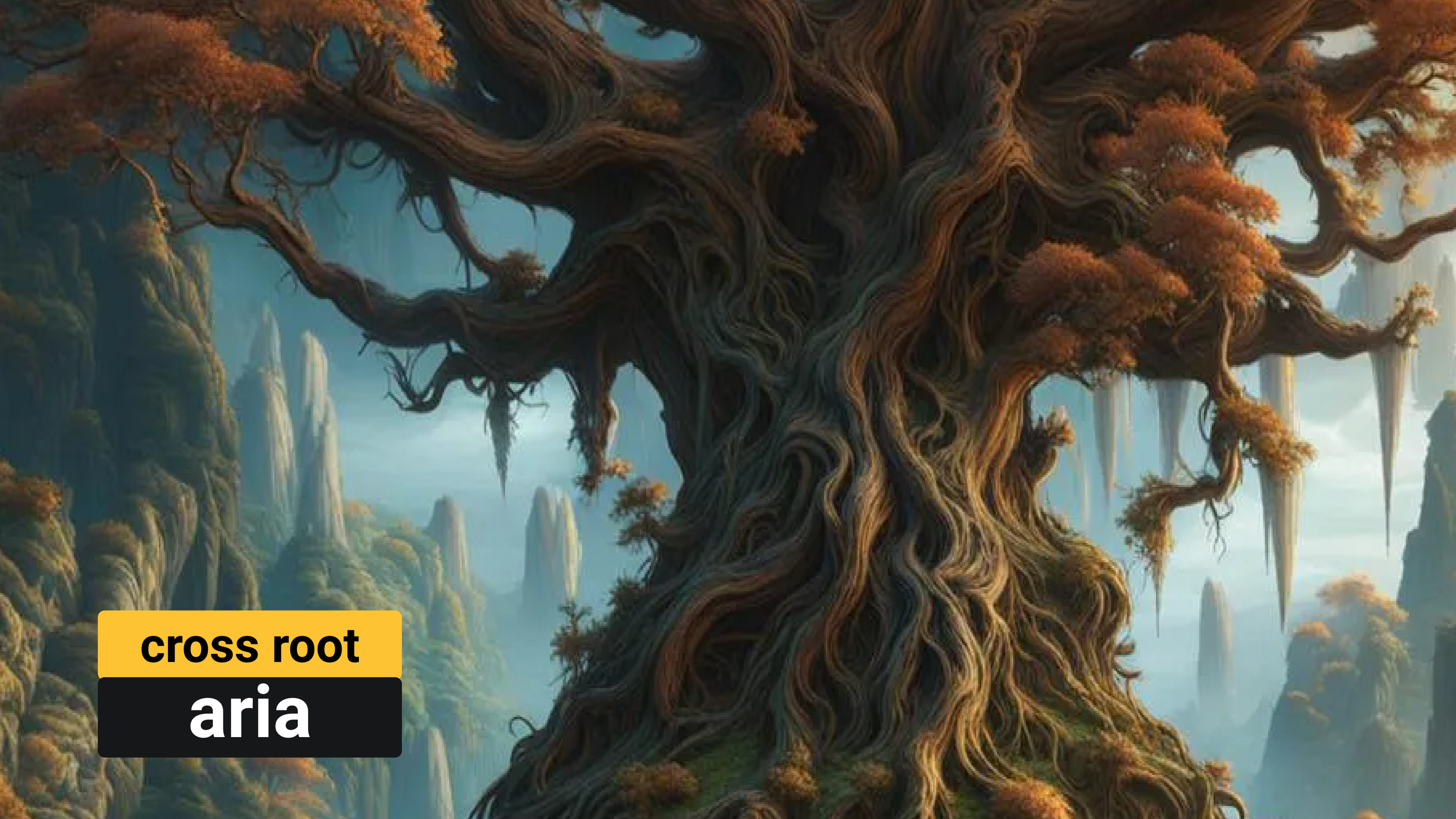

cross root
aria

Motivation

- lesser known and therefore underexposed topic

- not many resources on the internet
  probably because shadow dom is still a relatively new spec and many companies still use light dom centred tech stacks/frameworks

- lion heavily relies on shadow dom, so it is very important for us

cross root aria
table of contents

aria
a refresher

writing **accessible components** is

usually a matter of **applying the**

right attributes in the right context

(unfortunately, most sites today are written

without accessibility in mind...)

aria
a refresher

Let’s take **a form field as an example**

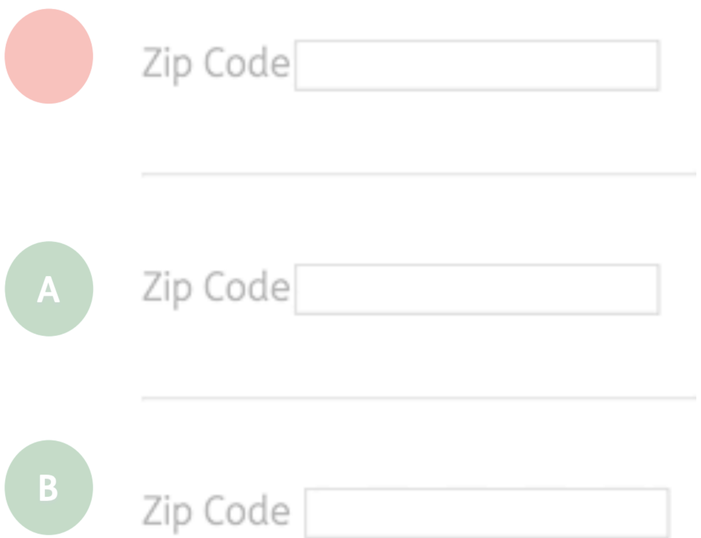
to see what the right application of

aria attributes can mean for end

users (screen reader users in

particular)...

aria
a refresher




```
<label>Zip Code</label>
<input>

<label for="zipcode">Zip Code</label>
<input id="zipcode">
```

> switch to presentation

“theory of forms”...

Problem

IDREFS cannot express relationships

between different DOM trees, c.q.

different shadow roots

cross root aria
the problem

one shadow root

```
<label for="zipcode">Zip Code</label>
<input id="zipcode">
```

two shadow roots

IDREFS are not working

```
<label for="zipcode">Zip Code</label>
<my-input>
  #shadowroot
  <input id="zipcode"/>
</my-input>
```

cross root aria
the problem

in complex scenarios these dom trees can grow quite large and complex...Think of:

- multiple nested roots
- sibling roots

cross root aria
our solution

Possible solutions

- abandon shadow dom
- copy nodes across shadow roots
- leverage light dom

https://github.com/leobalter/cross-root-aria-delegation/blob/main/explainer.md

cross root aria
our solution

Requirements

- shadow dom encapsulation
  (no style/dom leaks, best api via slots)
- cleanest/best performance
  (no mutation observers needed for id references)
- respects platform features (like implicit form submission, registration)
- slots provided by the consumer are respected
  (not moved around into shadow dom, causing potential styling issues)

Our solution

Leverage the light dom: it meets all mentioned requirements

We mainly do this via our **SlotMixin**:

https://lion-web.netlify.app/fundamentals/systems/core/slotmixin/

cross root aria
our solution

refresher: how

slots work

(content projection)

code from developer

```
<my-input>
  <label slot="label">Zip <b>Code</b></label>
</my-input>
```

outcome in browser

```
<my-input>
  <label slot="label">Zip Code</label>
  #shadowroot
  <slot name="label"/></slot>
</my-input>
```

cross root aria
our solution

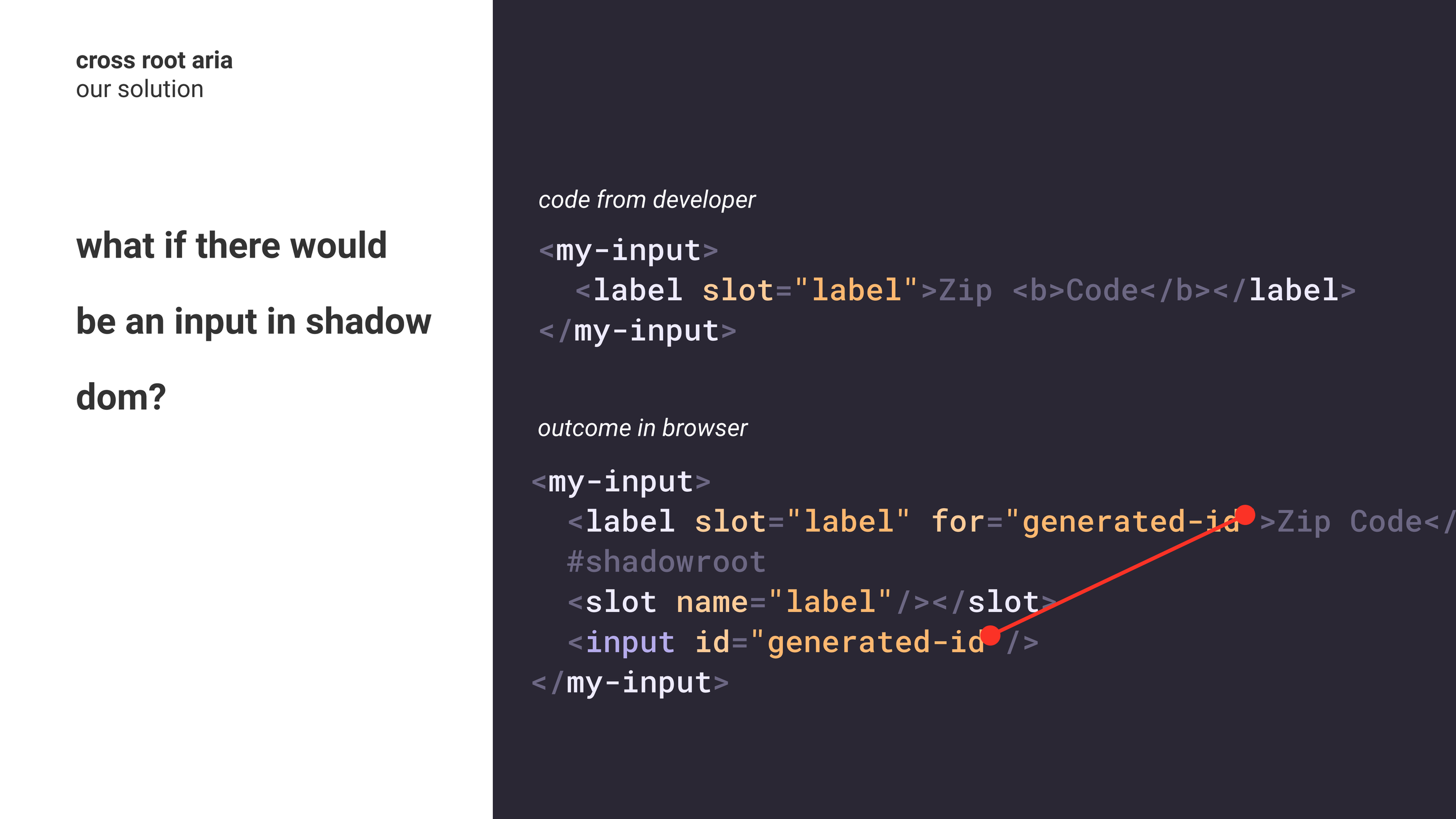
what if there would be an input in shadow dom?

code from developer

```
<my-input>
  <label slot="label">Zip <b>Code</b></label>
</my-input>
```

outcome in browser

```
<my-input>
  <label slot="label" for="generated-id">Zip Code</
  #shadowroot
  <slot name="label"/></slot>
  <input id="generated-id"/>
</my-input>
```



cross root aria
our solution

how SlotMixin solves

this with private slots

code from developer

```
<my-input>
  <label slot="label">Zip <b>Code</b></label>
</my-input>
```

outcome in browser

```
<my-input>
  <label slot="label" for="generated-id">Zip Code</
  <input slot="_input" id="generated-id"/>
  #shadowroot
  <slot name="label"/></slot>
  <slot name="_input"/></slot>
</my-input>
```

cross root aria
our solution

Sometimes we need

style encapsulation

code from developer

```
<my-input>
  <label slot="label">Zip <b>Code</b></label>
</my-input>
```

outcome in browser

```
<my-input>
  <label slot="label" for="generated-id">Zip Code</
  <my-styled-input role="textbox" slot="_input"
     id="generated-id"/>
  </my-styled-input>
  #shadowroot
  <slot name="label"/></slot>
  <slot name="_input"/></slot>
</my-input>
```

Spec proposals

- many proposals exist in different w3c working groups
- discussed at **TPAC** of september 2023

https://eisenbergeffect.medium.com/web-components-at-tpac-2023-f6da57519eb9

Candidate proposals at TPAC

- exportids

  https://github.com/WICG/aom/blob/gh-pages/exportid-explainer.md

- semantic delegate

  https://github.com/alice/aom/blob/gh-pages/semantic-delegate.md

- looking for best of both worlds

cross root aria
proposals

exportids

simple use case

```
<label for="zipcode::id(inner-zipcode)">Zip code</
<my-input id="zipcode">
  #shadowRoot
   | <input id="inner-zipcode" exportid />
</my-input>
```

cross root aria
proposals

exportids

forwardids

how to get

deeper

elements?

```
<x-address id="address">
  #shadowroot
   | <slot name="street-address-label"></slot>
   | <x-input id="street" forwardids="real-input: street-input">
   |   #shadowroot
   |    | <input id="real-input" exportid/>
   | </x-input>
   | <slot name="city-label"></slot>
   | <x-input id="city" forwardids="real-input: city-input">
   |   #shadowroot
   |    | <input id="real-input" exportid/>
   | </x-input>
  #/shadowroot
   <label for="address::id(street-input)">Street address:</label>
   <label for="address::id(city-input)">City:</label>
</x-address>
```

cross root aria
proposals

exportids

useids

how to access

sibling roots?

```
<x-label useids="label-for: gender::id(real-input)">
  #shadowroot
   | <label for=":host::id(label-for)">Gender</label>
</x-label>
<x-input id="gender">
  #shadowroot
   | <input id="real-input" exportid />
</x-input>
```

cross root aria
proposals

semantic delegate

- simpler
- edge cases

  hard

```
<label for="zipcode">Zip Code</label>
<fancy-input id="zipcode">
  <template shadowrootmode="open"
    shadowrootsemanticdelegate="actualinput">
    <input id="actualinput"/>
  </template>
</fancy-input>
```

Concluding

- reach consensus
- wicg -> w3c tpac -> whatwg -> browser 1 -> interop / baseline -> all browsers
- in the meantime: we will keep **lion forwards compatible**