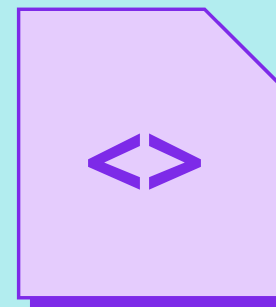
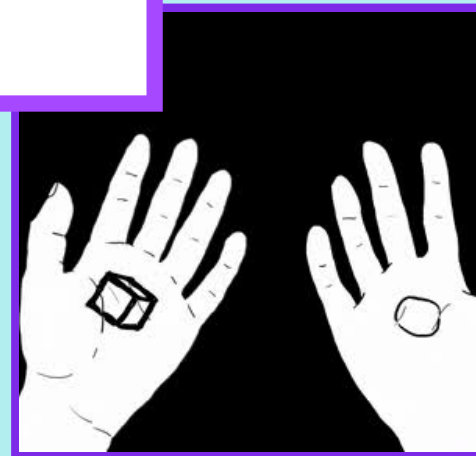


0101011011101?11EAD001011E101A01
01111000110101A01001001100100101
01P111110A0100A0011000E1010A1001
7011101011001P11?Y1100010011K1?0
0A111001000011111J000P1Y1010000A
1000000001001Y001100000001001010
111Y1101010111100001101100110011
1L1100100A00000110110111011111L
11010110110Y1111Y11U00?11A111011
0110?001001111010001000100110101
10110001F11011LA1110001A11010110
10101101110111F100110
101000111111111JPO1
001001011111000010101
111000101A000010P0000
11110101010000A0110?1
011001110111F11110101
001111101100111010010
000111100101001101000
111001100010100111000
11011001000101000U110
01001U1010E100111?111
100011100010010111000
00001100011101101000



Accessibility Part 4

Presented by
Mike Nam-Lee



4. Robust

Content must be robust enough that it can be interpreted reliably by a wide variety of user agents, including assistive technologies.

Parseable HTML

The HTML must be parseable. A lot of this comes for free as part of modern JS frameworks so I won't spend too long on this one.

Opening and closing tags and attributes must not be malformed.

```
<!-- Invalid -->
```

```
<p>Open tags must have an equivalent  
close tag
```

```
<!-- Valid -->
```

```
<p>Open tags must have an equivalent  
close tag</p>
```

```
<!-- Invalid -->
```

```
<input value="Self-closing tags must  
have a trailing slash">
```

```
<!-- Invalid -->
```

```
<input value="Self-closing tags must  
have a trailing slash"/>
```

Parseable HTML

Checks that won't be caught by modern JS frameworks:

- A doctype is defined
- No duplicate IDs
- Referenced IDs in aria-labelledby or htmlFor exist
- Only relevant attributes are added to tags

```
<!-- All HTML pages must start with -->  
<!DOCTYPE html>
```

```
<!-- Be careful of not defining multiple  
IDs -->
```

```
function Component() {  
  return <div id="componentId"/>;  
}
```

```
function Component2() {  
  return <><Component/><Component/></>;  
}
```

```
<!-- Instead, add some prefix to  
guarantee uniqueness -->
```

Elements with Roles

Interactive elements must either have a role or use a special tag that implies a role. As long as you're using buttons, inputs, and links, this should not be a problem. If you define your own custom interactive elements (e.g. anything with an `onMouseDown`, `onMouseUp`, `onClick`), you must add a role.

Any roles that you do add must be appropriate for the UX.

```
<!-- This is invalid -->  
<div onClick={...}>Update</div>  
  
<!-- This is better -->  
<button onClick={...}>Update</button>
```



**And that's
the end of
content!**



Use accessibility tools

Apple

Apple's accessibility tools come out of the box, both for macOS and iOS. Please have a go at using VoiceOver:

- [Guide for VoiceOver on Mac](#)
- [Guide for VoiceOver on iPhone](#)

Chrome

For all computers including Windows and Linux, you can use the [Chrome Screen Reader](#).

Android

[TalkBack](#) is the Android screen reader.

Additional topics



- **Reduced motion**
 - [MDN article on prefers-reduced-motion](#)
- **Dark mode**
 - [Dark mode can also improve accessibility](#)
- **Audio descriptions**
 - [Example of engaging audio descriptions](#)



All the best!