An Uncomfortable Place

Can we keep everyone happy with web components?

Hannah Clarke

Spot the difference...

A bit of background

UI Engineer in Design System Team

Intapp

A bit of background

UI Engineer in design system team

Intapp

SaaS provider for a wide range of industries

Lots of products

Lots of **acquired** products

Different tech stacks

Different FE frameworks

A lot of opinions

Rebranding priorities

A bit of background

The Challenge

Can we build coded components that can work for our product teams, regardless of framework?

The Challenge

Could web components be the answer?

A bit of my background

(a disclaimer)

What are web components?

What are web components?

"a set of web platform APIs that allow you to create new custom, reusable, encapsulated HTML tags to use in web pages and web apps"

WEBCOMPONENTS.ORG

What are web components?

Custom Elements

```
<my-custom-button>Woo!</my-custom-button>
```

Shadow DOM

Encapsulates elements in a shadow root (prevents styles leaking in/out)

HTML Templates

Define reusable HTML patterns that can create UI when needed

ES Modules

Export/import your web components across files

What does all this mean...?

Framework-agnostic components built using existing browser standards

What does all this mean...?

Framework-agnostic components built using existing browser standards

What does all this mean...?

Framework-agnostic components built using existing browser standards

What does all this mean...?

Problem solved! Right?!

React doesn’t play nice.

Data flow mismatch

Doesn’t listen

We need wrappers...

Problem solved! Right?!

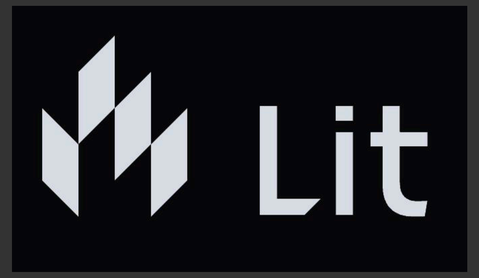
You don’t *need* a library, but...

Reduce writing boilerplate

Shadow DOM control

Framework wrappers

Other helpful tools



You don’t *need* a library, but...

Created for component libraries

TypeScript support

Output target libraries

Future proofing

You don’t *need* a library, but...

Problem solved! Right?!

Well... not exactly

The Next Challenge

Spot the difference...

Button +

Button +

Button +

The Next Challenge

Spot the difference...

Button +

```
<Button
    size="large"
    value="Button"
    variant="primary"
    rightIcon="AddIcon"
/>
```

😄

Button +

```
<uds-button
    label="Button"
    value=""
    name=""
    buttonType="button"
    variant="primary"
    size="large"
    rightIcon="add"
></uds-button>
```

Button +

```
<UdsButton
  label="Button"
  rightIcon="add"
/>
```

😠

The Next Challenge

The Next Challenge

React devs are loyal to their framework

Specific wrapper requirements

Output target library not necessarily providing everything we needed

A failed experiment?

(or, An Uncomfortable Place)

Let’s make it work

```tsx
import { type UdsButtonCustomEvent } from '@ids/web-components';
import {
    UdsButton as UdsButtonElement,
    defineCustomElement as defineUdsButton,
} from '@ids/web-components/dist/components/uds-button.js';
import type { EventName, StencilReactComponent } from '@stencil/react-output-target/runtime';
import { createComponent } from '@stencil/react-output-target/runtime';
import React from 'react';

type UdsButtonEvents = { onUdsRendered: EventName<UdsButtonCustomEvent<HTMLUdsButtonElement>> };

const UdsButton: StencilReactComponent<UdsButtonElement, UdsButtonEvents> =
    /*@__PURE__*/ createComponent<UdsButtonElement, UdsButtonEvents>({
        tagName: 'uds-button',
        elementClass: UdsButtonElement,
        // @ts-ignore - React type of Stencil Output Target may differ from the React version used in the project,
        react: React,
        events: { onUdsRendered: 'udsRendered' } as UdsButtonEvents,
        defineCustomElement: defineUdsButton,
    });

export default UdsButton;
```

Let’s make it work

We’ve got a script for that

```tsx
/**
 * This file was automatically generated by the UDS React post-build script.
 * Changes to this file may cause incorrect behavior and will be lost if the code is regenerated.
 */

import { forwardRef, memo } from 'react';
import UdsButton from './UdsButton';
import { UdsButton as UdsButtonProps } from './UdsButton.types';

const UdsButtonTemplate = forwardRef<HTMLUdsButtonElement, UdsButtonProps>(
    (
        { badgeRounded, buttonType, content, disabled, iconOnly, label, size, variant, ...rest },
        ref
    ) => {
        return (
            <UdsButton
                ref={ref}
                badgeRounded={badgeRounded ?? false}
                buttonType={buttonType ?? 'button'}
                content={content ?? ''}
                disabled={disabled ?? false}
                iconOnly={iconOnly ?? false}
                label={label ?? ''}
                size={size ?? 'large'}
                variant={variant ?? 'primary'}
                {...rest}
            />
        );
    }
);

const UdsButtonReact = memo(UdsButtonTemplate);
UdsButtonReact.displayName = 'UdsButton';

export default UdsButtonReact;
```

Let’s make it work

We've got a script for that

Need component types files?

Need to pass specific info to the wrapper?

Substantial changes to React?

Yep, we can fix it

Let's make it work

What does it mean in practice?

Write components once

Provide for teams using other frameworks

Component packages are always in sync

What does it mean in practice?

Still write tests and Stories in each package

Components sometimes misbehave

Stencil sometimes misbehaves

Contribution can be tricky

What does it mean in practice?

As a team of only three UI Engineers, if we had to write and maintain components in every framework, it would be an insurmountable task.

What does it mean in practice?

Where are we now?

Web components and React packages

Adding support for SSR

Working with teams to add Angular

Teams are adopting our components regularly

Where are we now?

Can you keep everyone happy?

Spot the difference...

Seems like a win.

Thanks!

Hannah Clarke