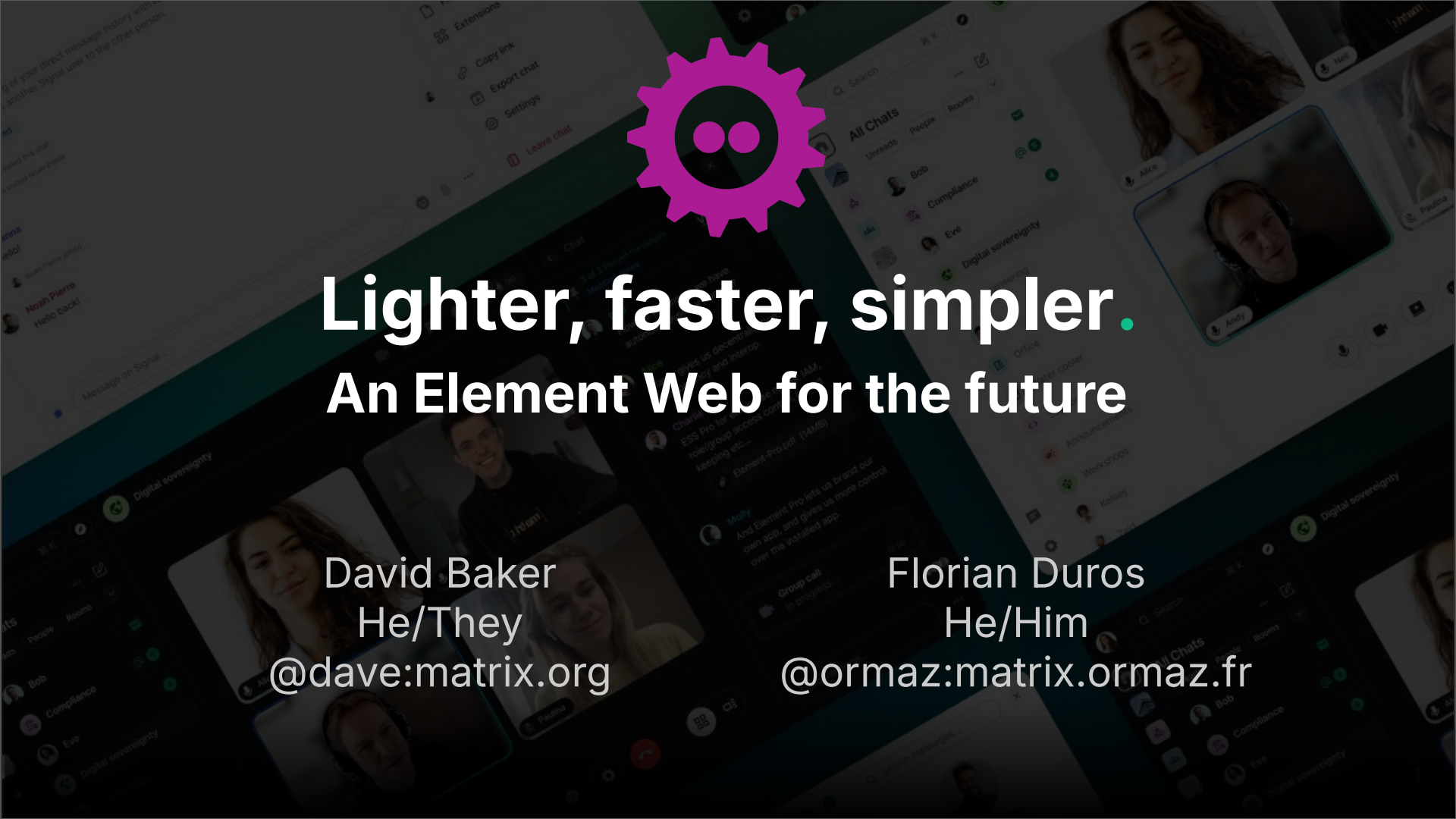


Lighter, faster, simpler.

An Element Web for the future

David Baker
He/They
@dave:matrix.org

Florian Duros
He/Him
@ormaz:matrix.ormaz.fr

Agenda

- Intro
- The Journey to Element Web today
- The Destination: Matrix Rust SDK
- The Map: How do we get there?
- Deep dive: MVVM and shared components
- Demos!
- Questions

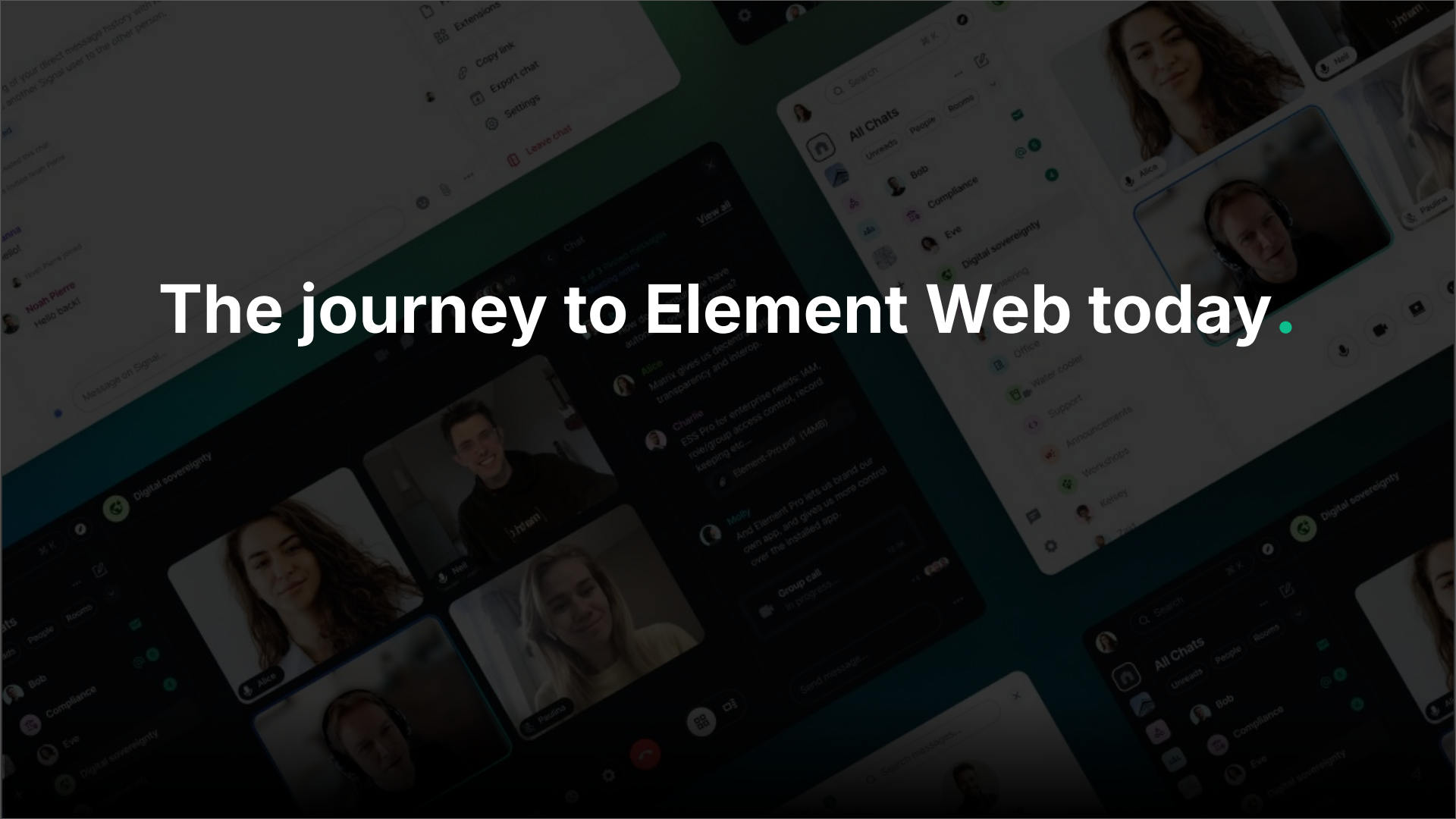
Speakers

David Baker
Staff Software Engineer, Element
Matrix Spec Core Team

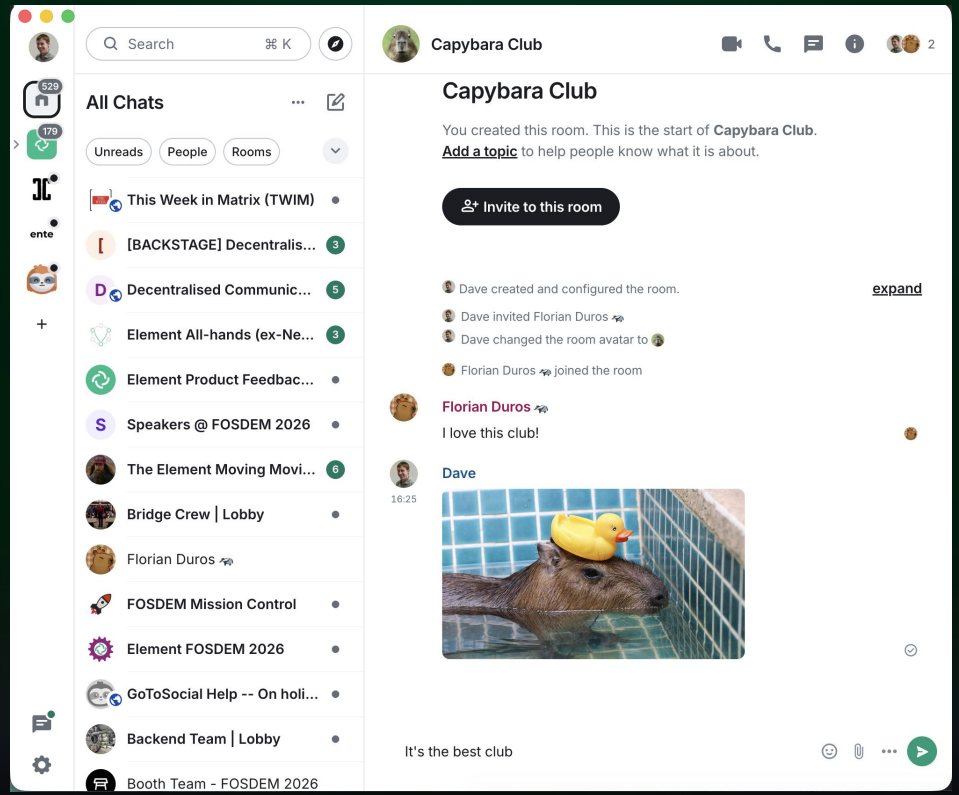
@dave:matrix.org

Florian Duros
Senior Software Engineer, Element

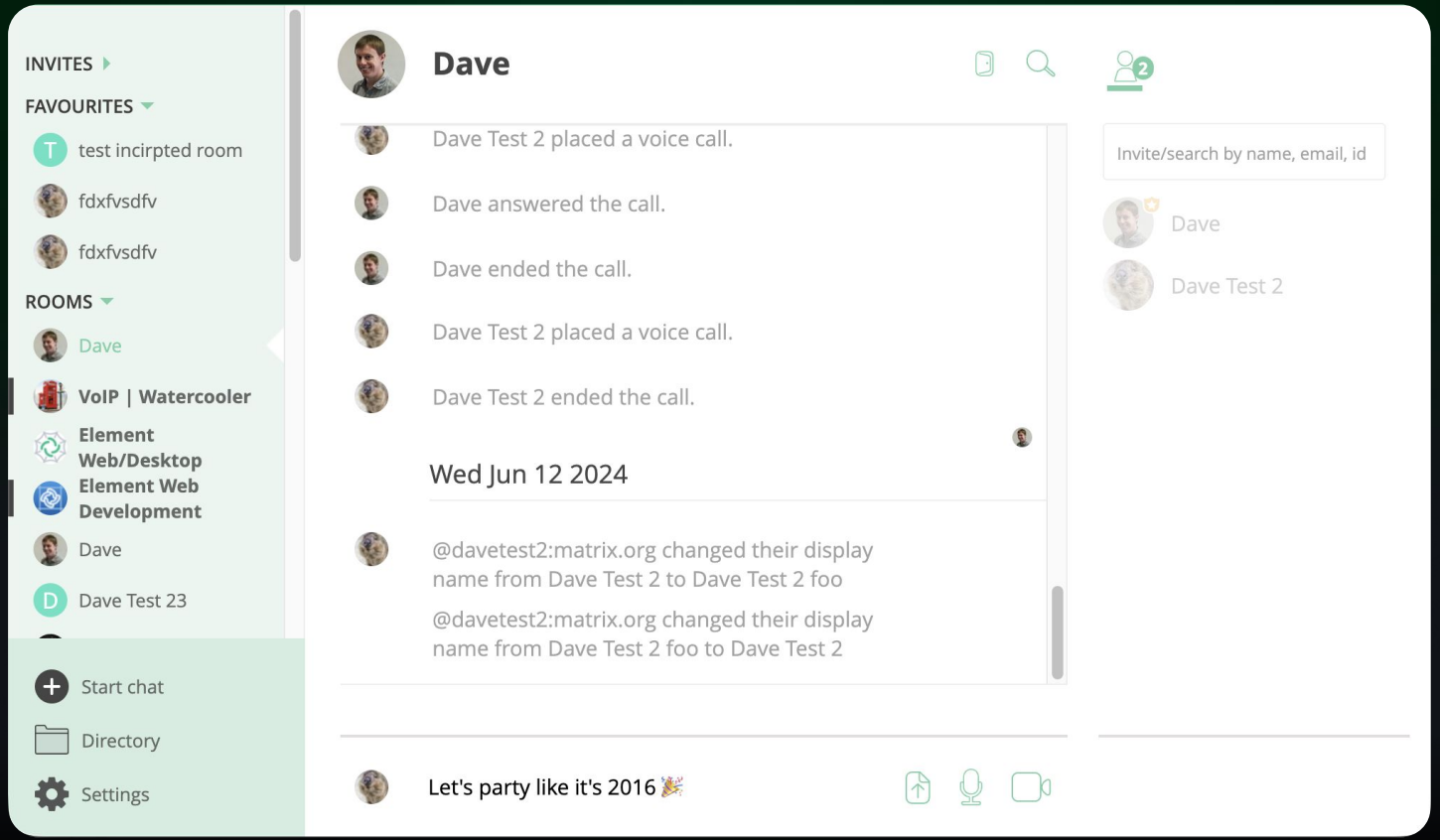
@ormaz:matrix.ormaz.fr
@florianduros:element.io


The journey to Element Web today.

A React client for Matrix



…the early days!



A React client for Matrix

- Our first Matrix client built on React
- Started in 2015
- Built on (and alongside) matrix-js-sdk
- Most data / state stored by js-sdk
- Nominal Flux pattern (pass data down, dispatch up)
- Very fast... at first!

Reusable Parts

- Most code actually in a separate package: Matrix React SDK
- Intended to be a reusable SDK to build a Matrix app in React
- Expectation vs. reality...
- React SDK components weren't really re-usable in isolation
- Quite a lot of logic in components

Organic Growth

- More & more features have been added over the years
- As with all large projects, has accumulated technical debt as it’s grown
- Technology and coding standards have changed in the last decade!
- Open source means contributions need managing if they’re to maintain the same code style.

Leading Matrix on the Web

- The Element Web codebase is one of the most feature-complete Matrix clients in the ecosystem and is key to many organisations using Matrix.
- Including forks, the most widely deployed Matrix client




Setting the Standard

If Matrix, **and decentralised chat as a whole**, is to succeed widely, it needs a web client that's a **fast** and **reliable** as the closed source, centralised competitors.


The Destination.

Matrix Rust SDK

- Both Element mobile apps now powered by Matrix Rust SDK
- Rewritten from scratch:
  - Faster
  - “Sliding sync” (instant launch)
  - Memory efficient
  - E2E built in from the very start
  - Scalable design
  - Shared models with Ruma
  - Cross-platform consistency

Element Web ‘X’

- Can we have Element Web backed by the same SDK as the mobile clients?
- Run the Rust SDK via WebAssembly
- No longer need to write each feature twice
- Should lead to higher quality apps
- Already uses the crypto part of Matrix Rust SDK


The Map.

Avoiding a Rewrite

- The Element X mobile apps are vastly improved having been rewritten on top of the Rust SDK
- ...but the migration is still not complete.
- How can we iterate on Element Web, migrate to Matrix Rust SDK and improve the codebase in the process?

Shared Components

- Reusable, but opinionated: designed for Element.
- Can be used in Element Web Modules too!
- Agnostic of any Matrix SDK
- They are UI code and nothing more

- A few small components done so far
- **In progress:** Room List, Timeline tiles

Shared Components: Medium Term

- Right panel (member list, room info etc.)
- Space panel
- Login & registration views

Plus, **prove these components are reusable by reusing them in another app!**

Shared Components: Long Term

Decoupled, well-defined interfaces: easier to replace the code that drives them.

We can then start to migrate to the Rust SDK, now that the view level is SDK-agnostic.

MVVM and Shared Components.

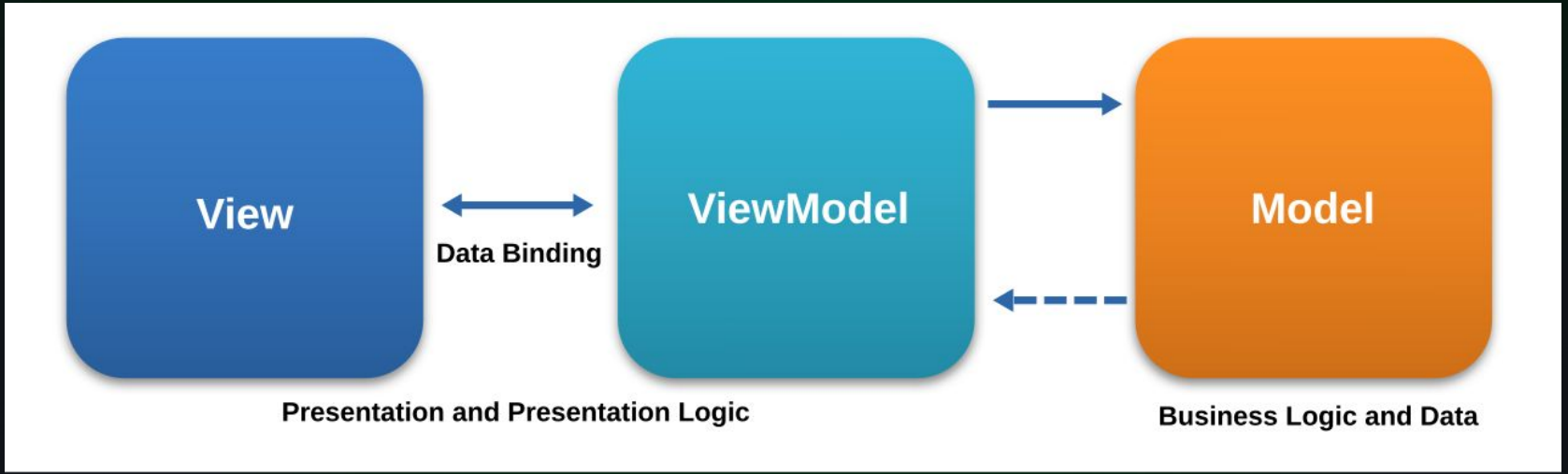
Technical detail

Florian Duros

MVVM

- Model–view–viewmodel (MVVM) facilitates the separation of the development of a **graphical user interface** from the development of **the business logic** such that the view is not dependent upon any specific model platform. *Wikipedia*

MVVM



Wikipedia

MVVM in Shared Components

- The **View** lives in Shared Components
- Shared Components defines the interface
  - The expected data (snapshot)
  - The actions to call on the view model
- The **Viewmodel** is provided by the application
  - Element Web
  - EW module
  - Aurora
- Internally **Viewmodel**s use `React.syncExternalStore`

MVVM in Shared Components

```tsx
export interface MyCompViewSnapshot {
  label: string;
}

export interface MyCompSearchViewActions {
  onClick: MouseEventHandler<HTMLButtonElement>;
}

export type MyCompViewModel = ViewModel<MyCompViewSnapshot> &
  MyCompSearchViewActions

interface MyCompViewProps {
  vm: MyCompViewModel;
}

export function MyCompView({ vm }: Readonly<MyCompViewProps>): JSX.Element {
  const { label } = useViewModel(vm)

  return (
    <button type="button" onClick={vm.onClick}>
      {label}
    </button>
  )
}
```

MVVM in Shared Components

```
interface Props {}

class MyCompViewModel extends BaseViewModel<MyCompViewSnapshot, Props> implements
MyCompViewModelInterface {
    public constructor(props: Props) {
        super(props, {
            label: "Click Me",
        });
    }

    public onClick: () => void = () => {
        console.log("Button clicked");
    };
}
```

Shared Components

- We completely decouple Shared Components from the app logic from the start
  - No dependencies from Element Web
- We re-use Shared Components in other apps as early as possible
- **Most of all:** new UI or UI refactoring is done in Shared Components
- We also modernized our tooling! Vite, vitest, storybook, CSS module...


Demos.

Aurora

https://github.com/element-hq/aurora

A hackathon project to see if we can build a web / desktop client using the Rust SDK.

- Initially desktop powered by Tauri
- Now runs on web using Rust SDK in Web Assembly
- Very basic, most components copied & pasted from Element Web... until now!


Questions?.