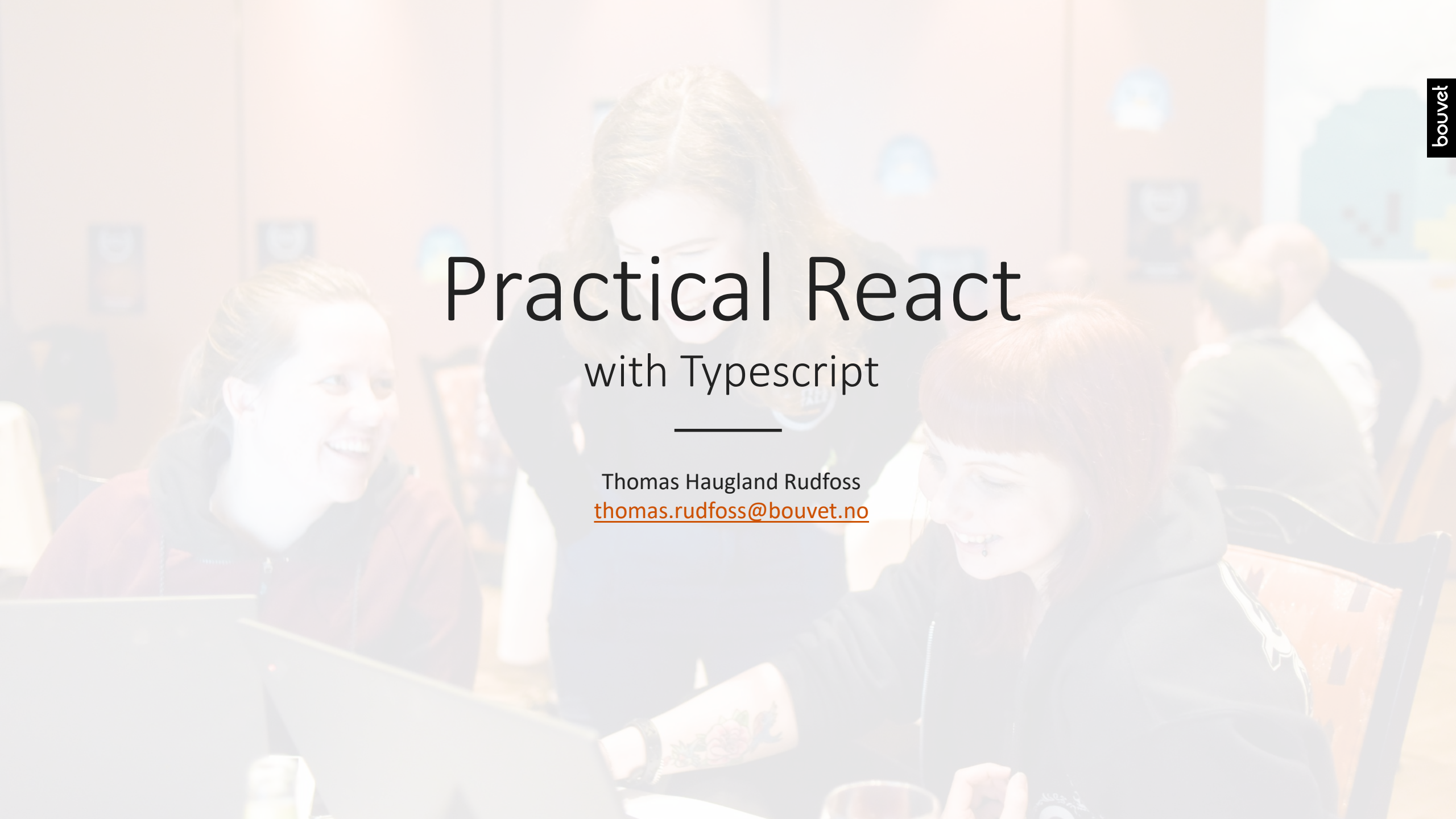


# Practical React

## with Typescript

Thomas Haugland Rudfoss
thomas.rudfoss@bouvet.no

bouvet

# Practical React with TypeScript

- Code repository
  https://github.com/rudfoss/cra-workshop

- Guest WiFi
  Network name: Bouvet Guest
  SMS “wifi” to 45 94 80 40

- Contact
  thomas.rudfoss@bouvet.no

bouvet

# React

> *A JavaScript library for building user interfaces*
>
> *- reactjs.org*

bouvet

# Agenda

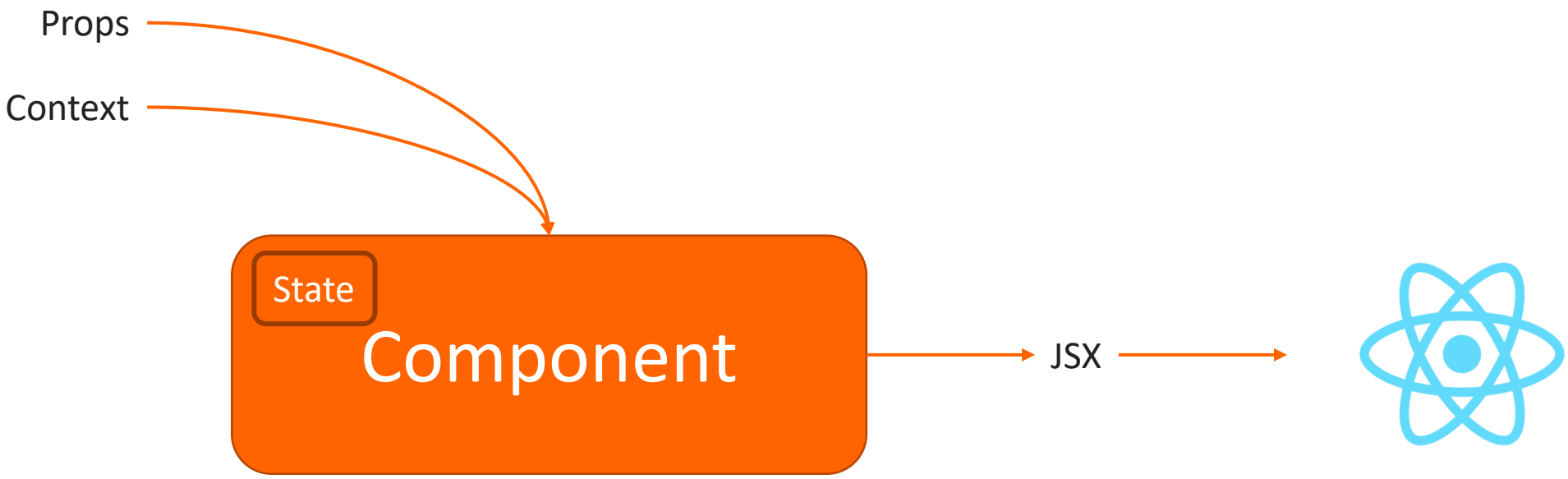
- Grunnleggende React
  - Komponenter og komponent-treet
  - JSX
  - Props/state
  - Flux-pattern
  - «Hoisting»
- Komponentstruktur
  - Livssyklus-hooks (useEffect)
  - State hooks (useState, useReducer)
  - Memoization (useMemo, useCallback, memo)
  - Composition
  - Kodestruktur, mappeinndeling, feature-folders
  - TypeScript med React
- Kodepatterns
  - Context (services)
  - Arbeide med immutable data
  - Ruting (med react-router)
  - Lasting av server-data (med React-Query)
  - Kodesplitting (bundle-splitting)
- Diskusjonstema
  - Mono-repository (fordeler, ulemper)
  - Global state (redux, zustand, annet?)
  - Rammeverk (NextJs, Create-React-App, NX Workspace)

bouvet

# Demo – Hello world

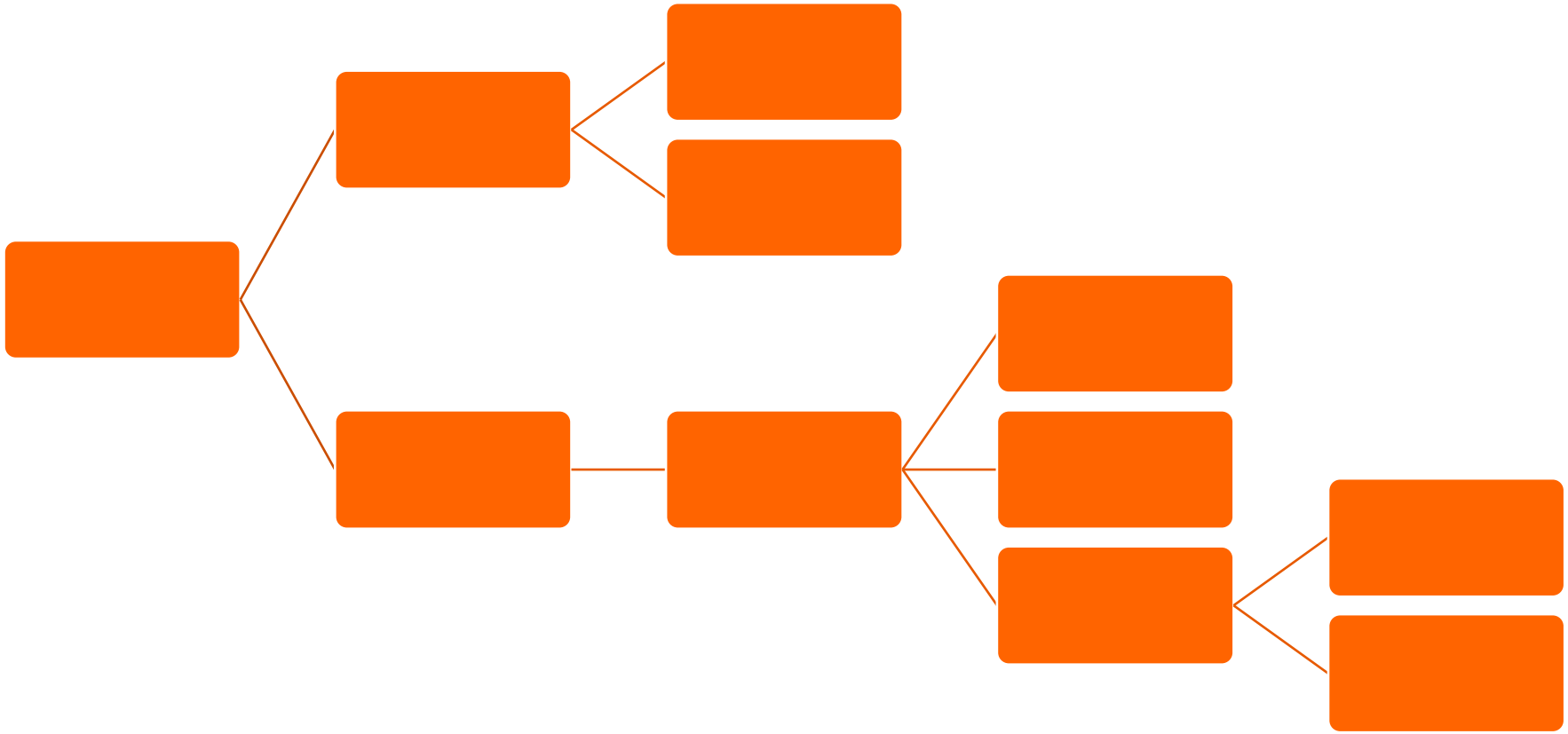
- Create a component that displays the text «Hello World» in an H1 tag.
- Use it in the application.

# The React component



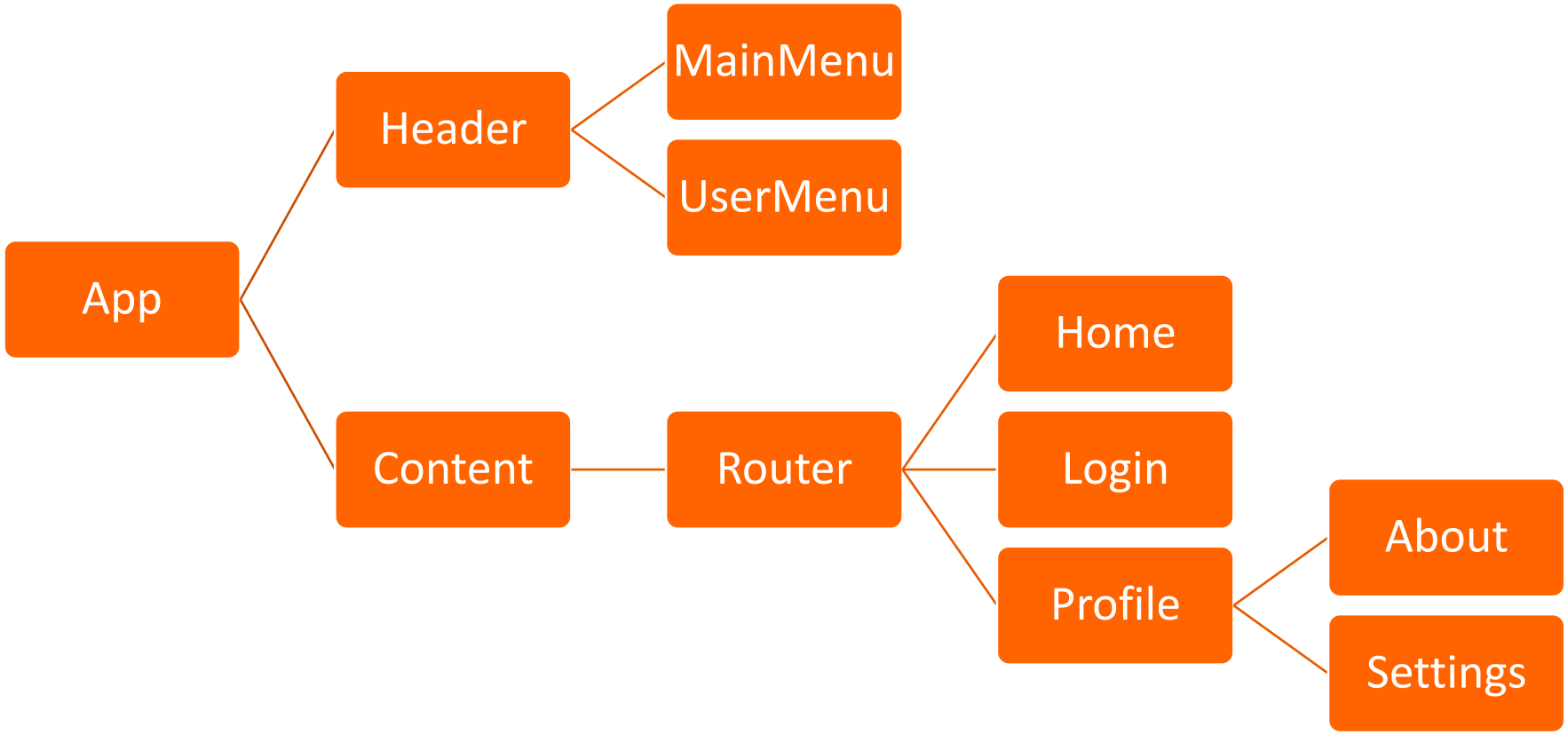

bouvet

# Component tree



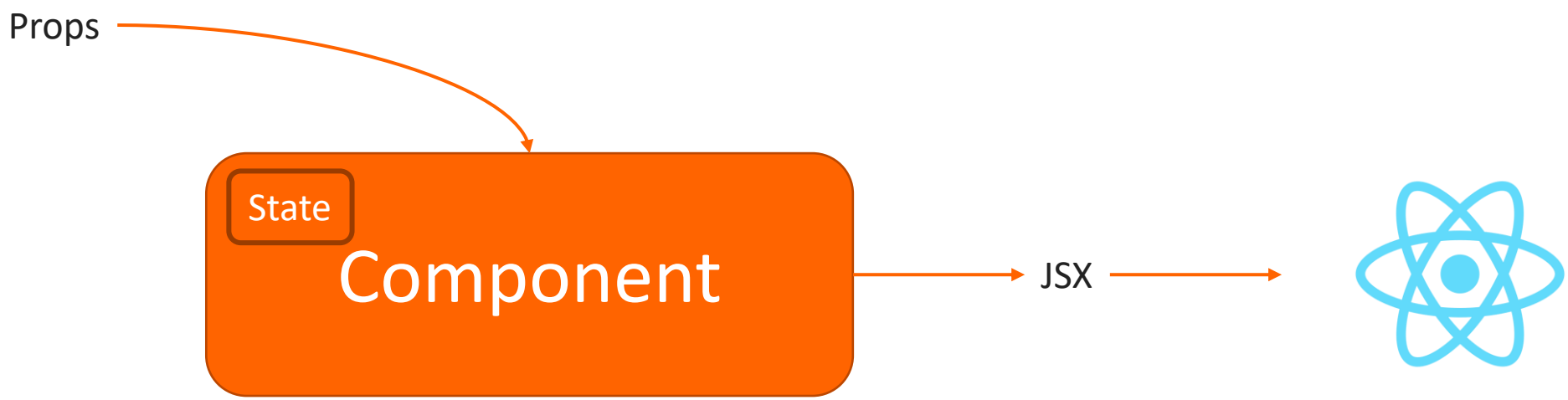
bouvet

# React app



# Props and state

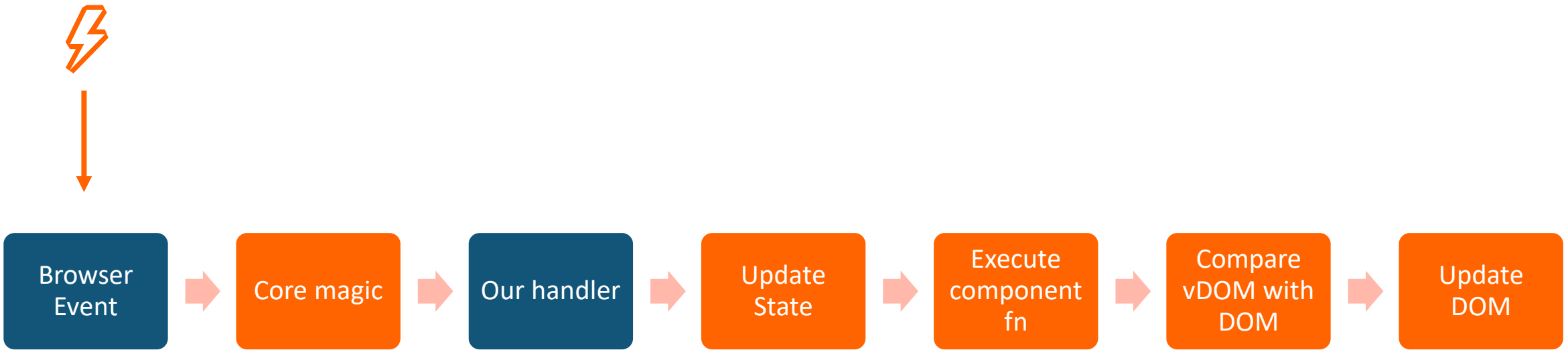
- Props: Incoming values (= arguments)
  - `<Comp prop="hello"/>`
- Children: Prop between opening and closing tags
  - `<Comp>children</Comp>`
- State: Internal values



# Demo – Heading

- Create a component that takes a text prop and renders it in an H1 tag.
- Provide another optional prop «asSpan» that takes a boolean and renders the text in a span instead of an H1 if set.

# React event loop*




bouvet

# Task – Incrementor

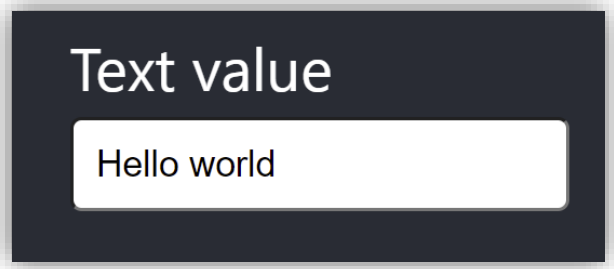
- Create a component with a button with a number as its text.
- The text increments by one when the button is clicked.
- When a limit is reached the button is disabled.
- Add a reset button that resets the counter to 0. Only show the reset button when the limit is reached.
- Props
  - limit – number – The maximum number the incrementor can reach before it is disabled
  - children – string – A text displayed once the limit is reached (optional with a default text)

# Task – TextInput

- Create a component that renders a proper text field with a label and a text input.
- Clicking the label should put focus in the text field
- Print the text you write under the component with the prefix: “You wrote [value]”
- Bonus: Style the component

```
<TextInput label="First name" />
```

```
<div>
  <label for="textField_0">Text value</label>
  <input id="textField_0" type="text">
</div>
```




bouvet

# Hoisting state

- Make components reusable
- Delegate responsibility of state management
- Be wary of perf (memoize)

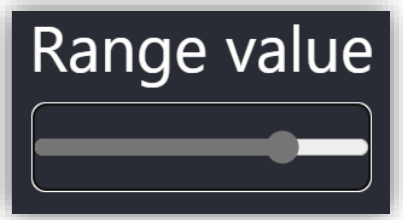
# Styling

- Style-prop
- CSS (imported)
- Styled-Components*
- …

bouvet

# Task – RangeInput

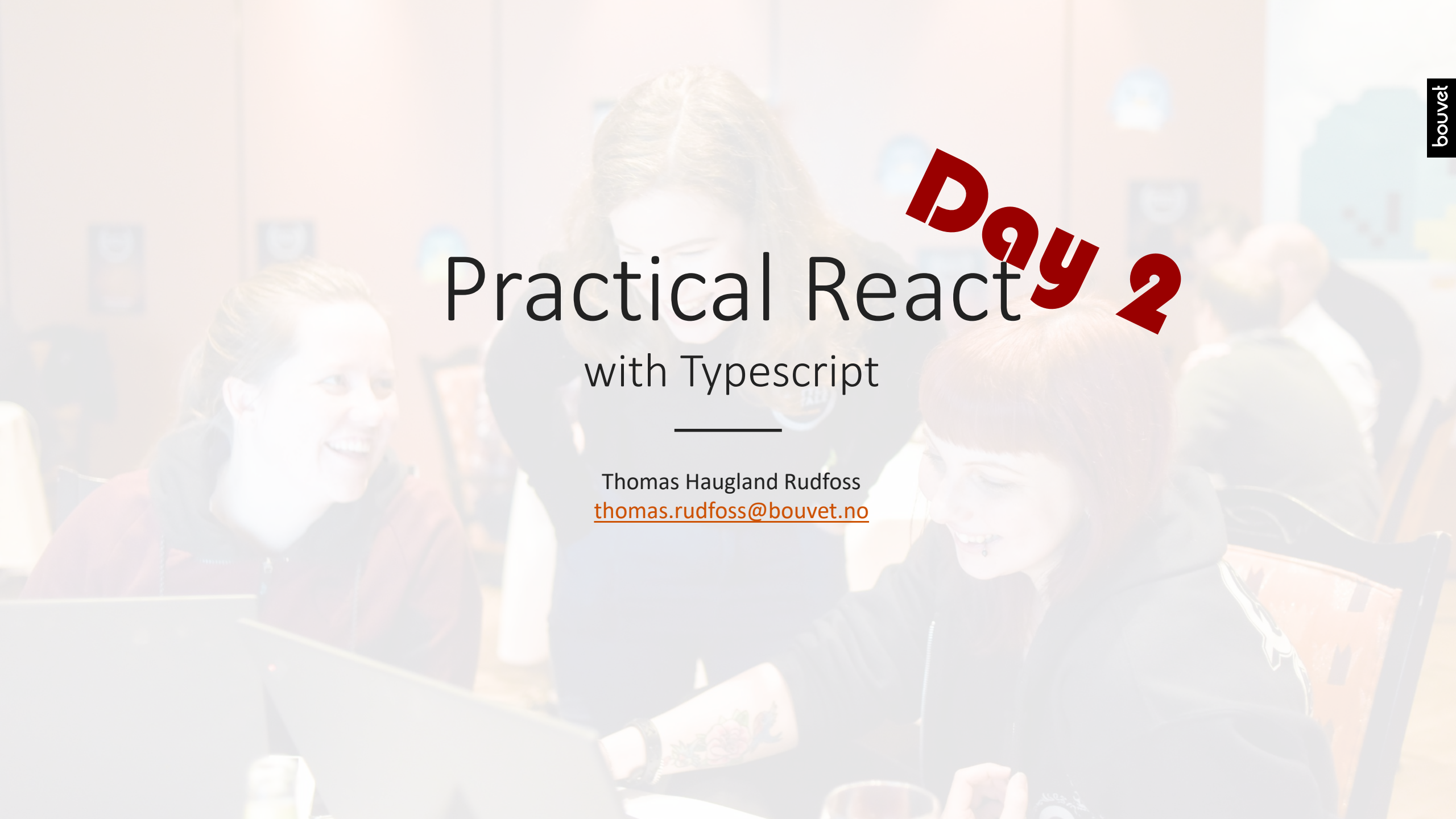
- Create a range input with a label and configurable min and max values.
- Give it some styling.
- Hoist the state so that it can be reused



# Task – Checkbox input

- Create a checkbox input with a label
- Give it some styling
- Hoist the state so that it can be reused

bouvet



# Practical React

Day 2

## with Typescript

Thomas Haugland Rudfoss
thomas.rudfoss@bouvet.no

bouvet

# Hoisting and performance

bouvet

# Loops

- Repeating components
- Key-property

bouvet

# Demo – Loop static data

- Download data from: https://dummyjson.com/users
- Create a component that lists this data

bouvet

# Task – Simple Todo

- Create a Todo component with a text field and a button.
- The button should only be enabled if there is text in the text field.
- When the button is clicked add the text to a list of todos that is displayed below the input. It also clears the text.
- Add styling as needed.
- Bonus: Pressing enter when the text field has focus should also add a todo.
- Bonus: Add a remove button to each todo item.

# Component composition

- Components can specify placeholders for other components.
- Control the “surroundings” of a component.

# Demo – Repeat

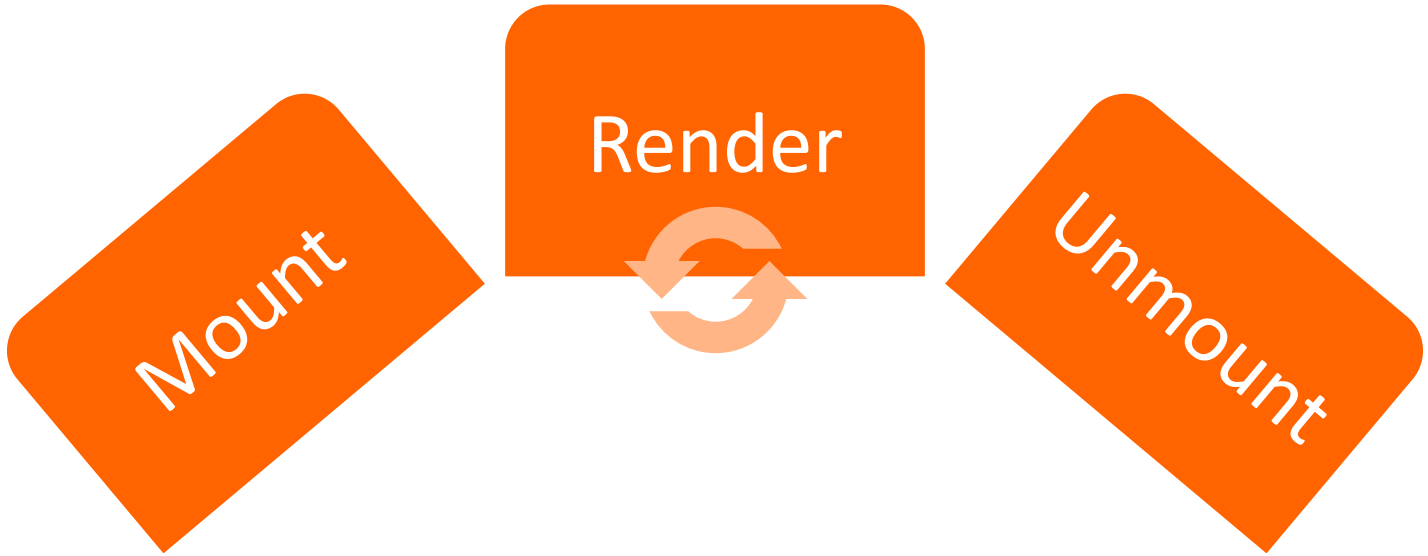
- Create a component that takes a single `React.ReactElement` as a child and a prop count of type number.
- Use RangeInput to control the number of times the child is repeated.

# Task – BigBorder

- Create a component that shows its children in a box with a red border.
- Control these style properties with range inputs:
  - border-width
  - padding
  - border-radius
- Use a checkbox to toggle whether the component renders its children or not.

# useEffect

- Run code that has a side effect on someting outside of Reacts control.
- Run code that depends on the lifecycle of a component.



# Demo – useEffect

- Create a component that logs a message to the console when mounted.
- Log another message when it is unmounted.

# Task – useEffect window title

- Create a component «PageTitle» with a text input.
- Once the component is mounted store the old window title and allow the user to change it using the text input.
- Once the component is unmounted restore the original title.

- Create another component that only renders the the PageTitle component if a checkbox is checked

# Demo – useWindowTitle custom hook

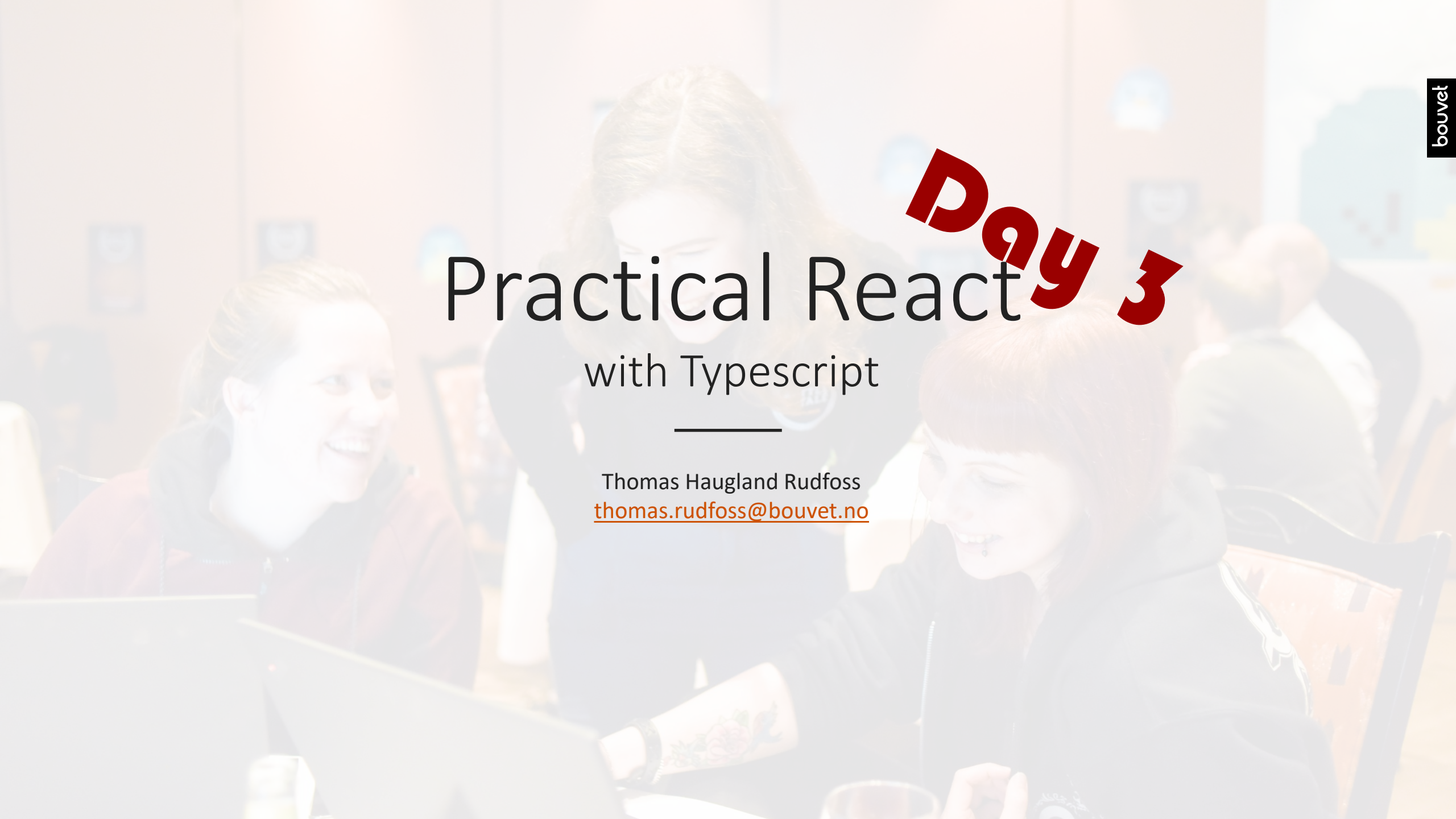
- Create a custom hook that allows setting, and optionally resetting the window title.
- It should respond to changes to the title parameter.

# Testing

- Create React App includes React Testing Library.
- Test the way a user would interact with the application.

# Code restructuring

- “Intent” folders
  - demos
  - tasks
  - ui
  - utils
  - data
  - …
- “API Facades”
- Alias paths



# Practical React

Day 3

## with Typescript

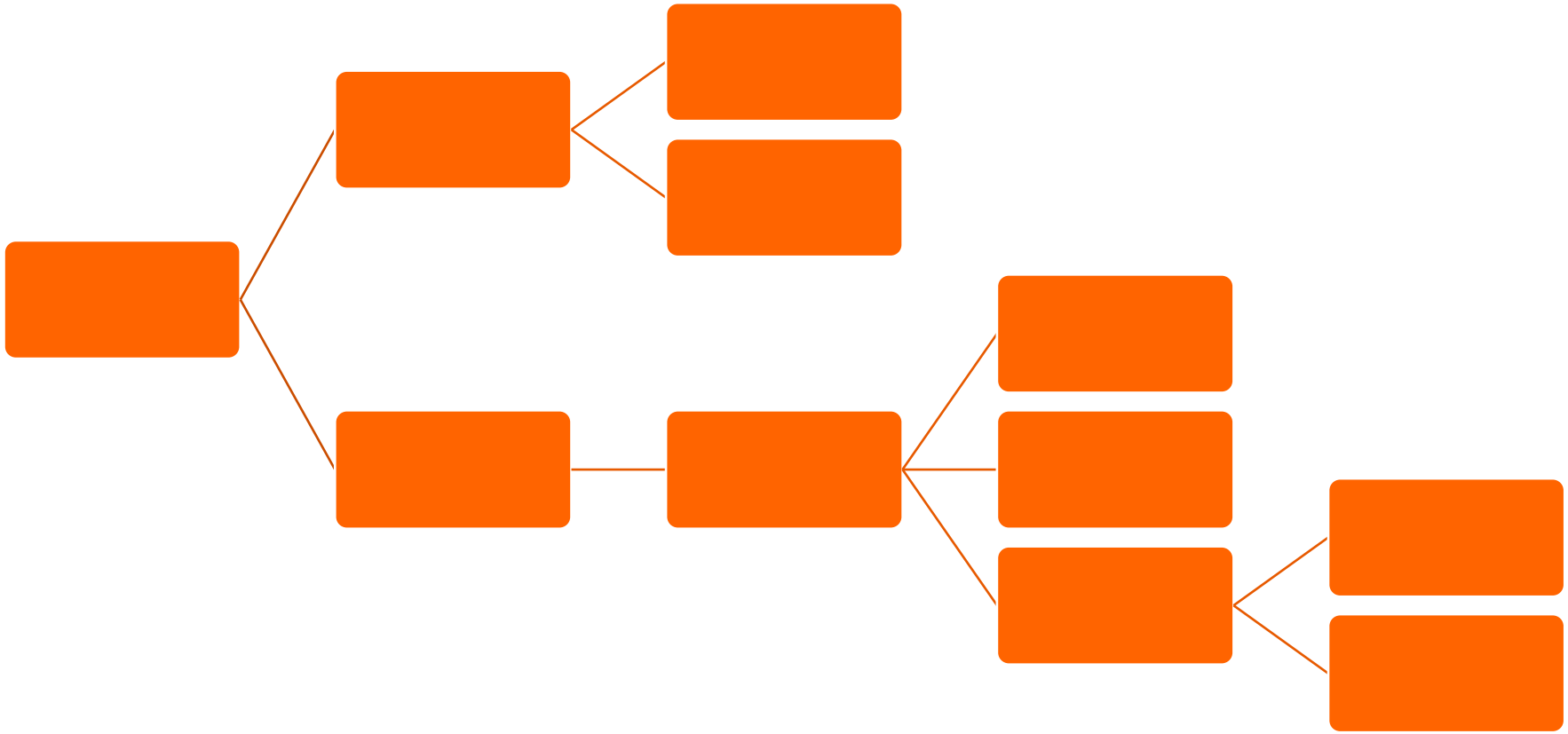
Thomas Haugland Rudfoss
thomas.rudfoss@bouvet.no

bouvet

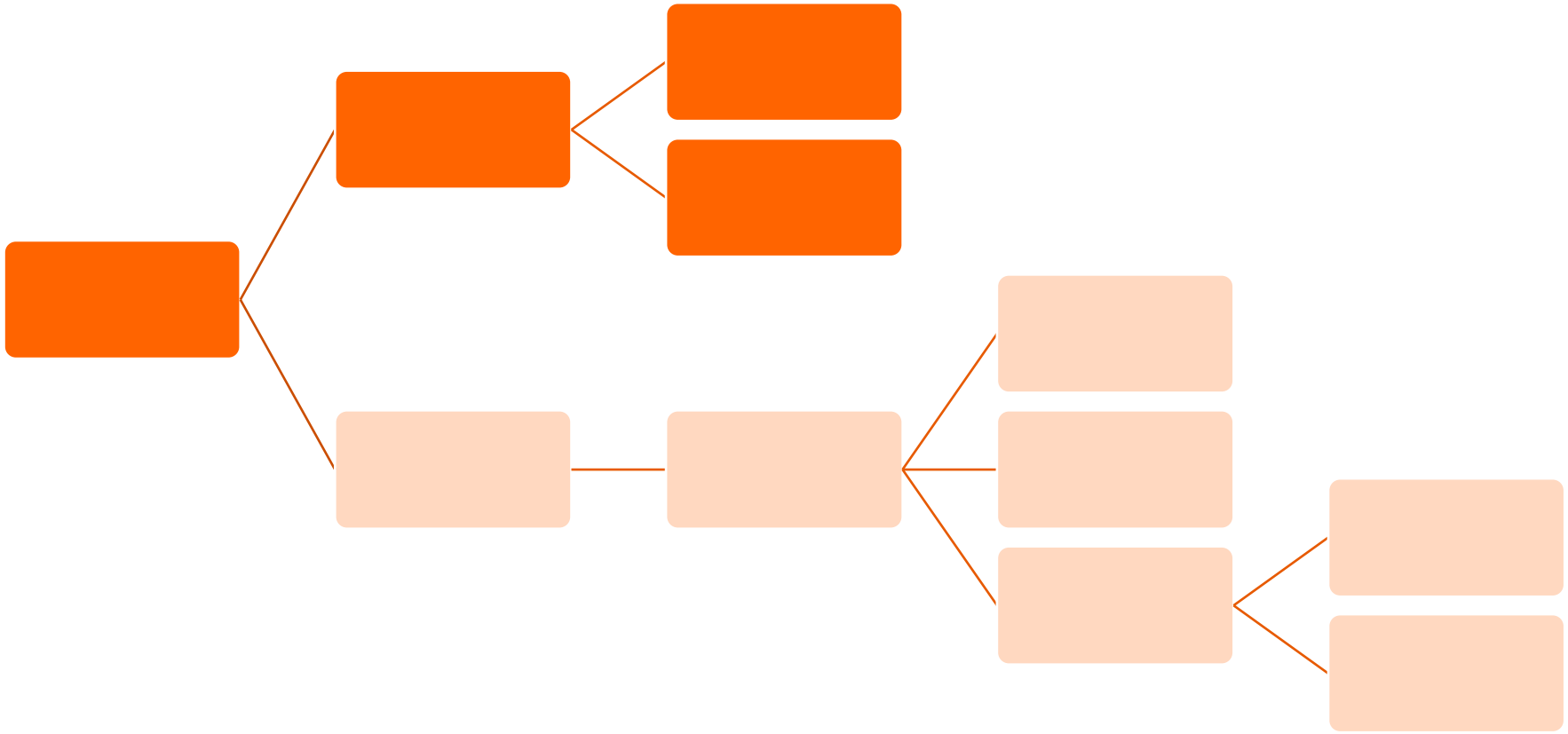
# Routing

- A route is a filter on a url «pattern»
- Filters can be applied anywhere
- Parameters can be extracted from the url
- The url does not have to come from the address bar

# Routing

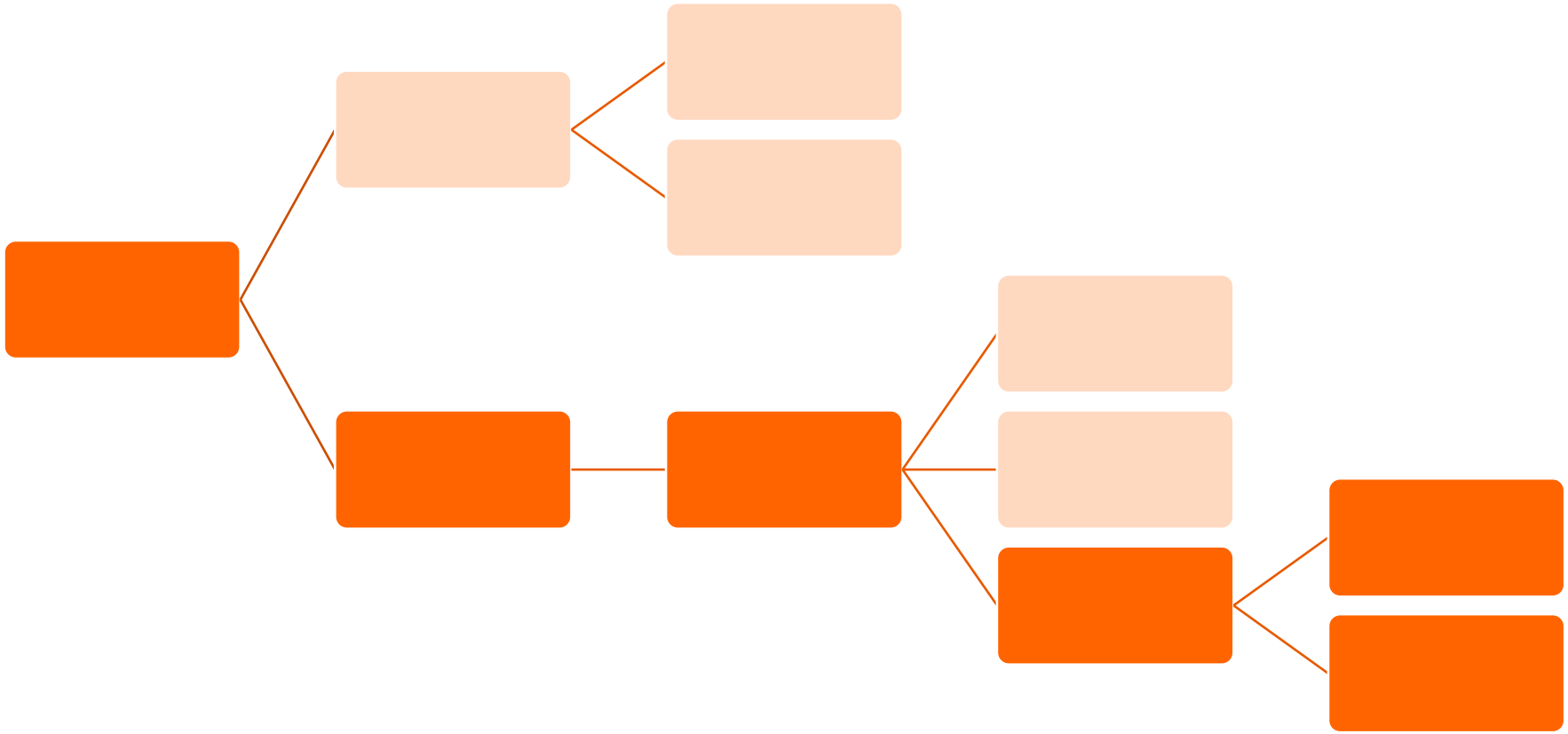


# Routing



bouvet

# Routing



# Demo – Set up router

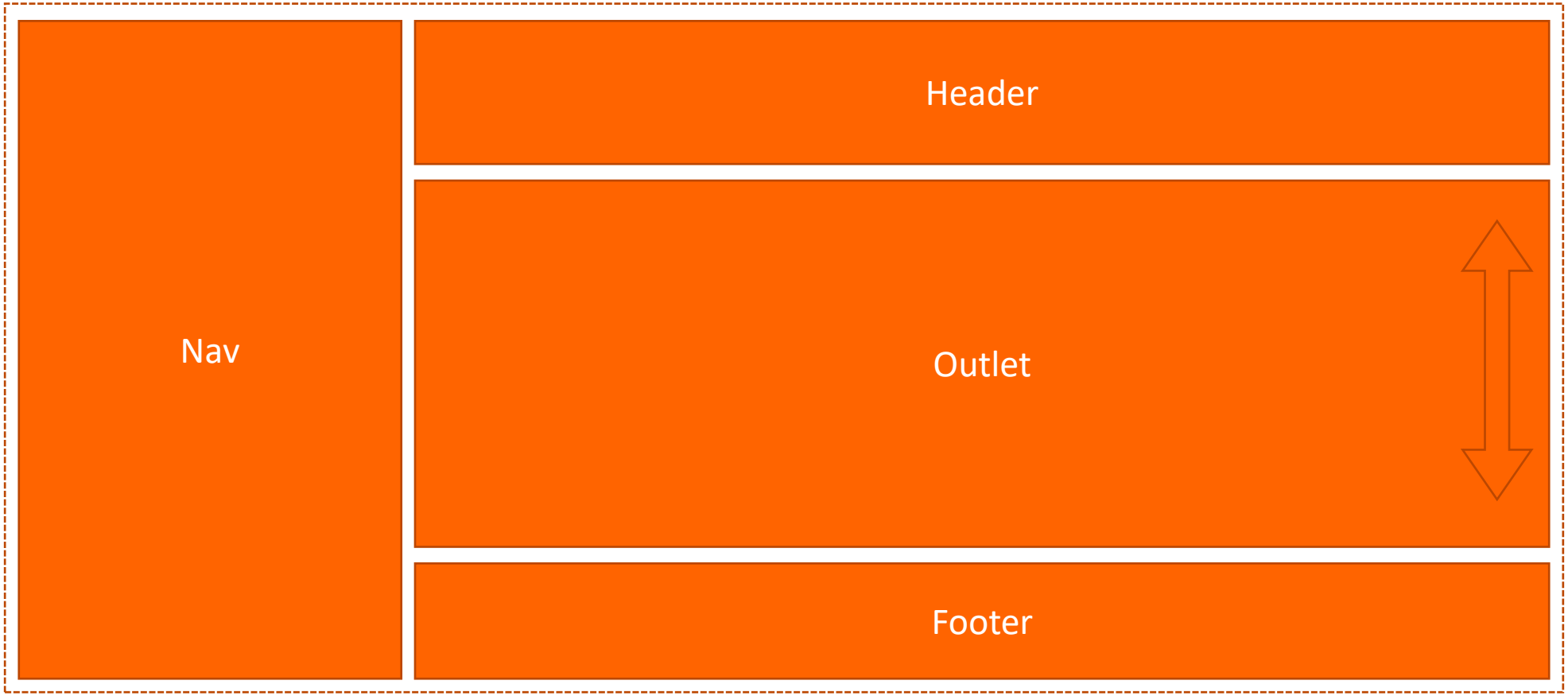
- Set up routing using react-router.
- Set up a simple layout.
- Create a «Not found» page with a «go back» button.
- Create a home page and a page for the Hello World and Repeat demos.
- Get the repeat count for the Repeat demo from a url parameter.

# Task – Nav

- Create a Nav component to hold navigation links.
- Create a separate page and url for each task we have completed and links to them to the Nav component.
- Create a «Home» link in Nav to return to the home page.

# Task – Create a layout

- Create this layout and use it as the main layout of the application



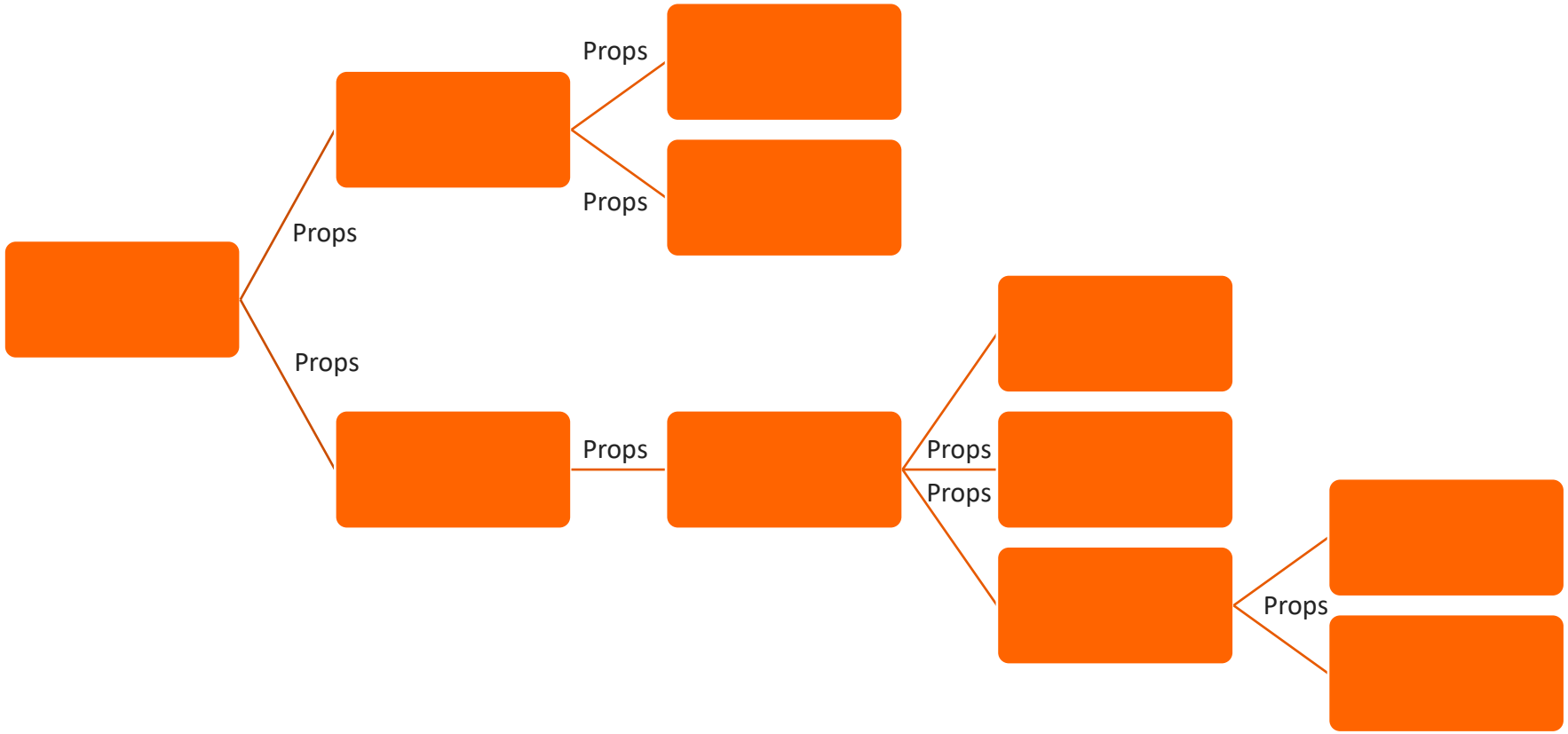
# Code splitting

- Split the application into smaller pieces.
- Load pieces only when needed (lazy-loading).

# Context

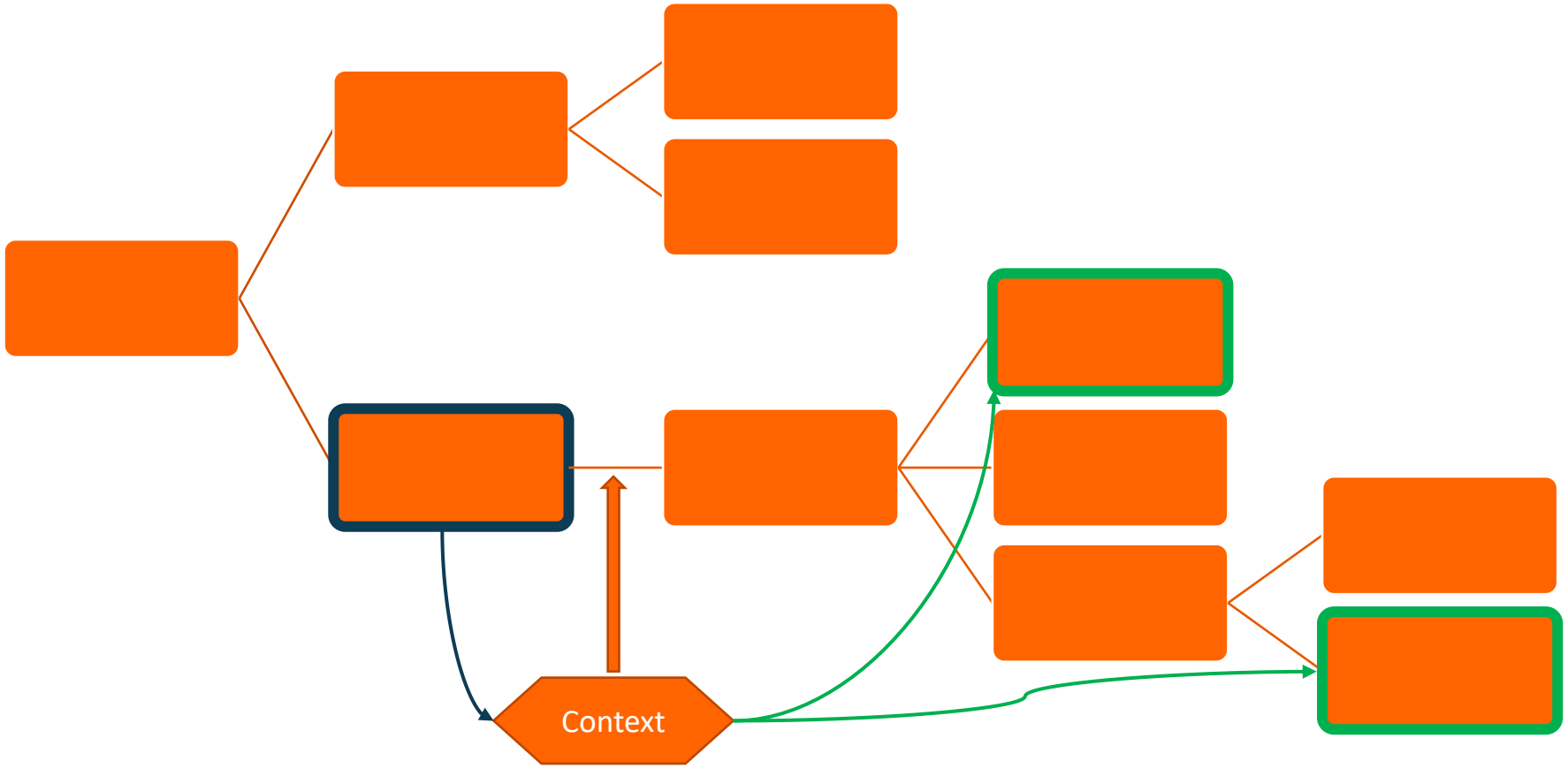
- «Pass props to any descendant»
- Component is aware of changes and updates accordingly
- “Dependency injection” (kind of)

# Props




bouvet

# Context




bouvet

# Demo – Create settings context

- Create a page for controlling application settings.
- Create a context for holding these settings.
- Create a range input that controls the width of the navigation in pixels.

# Task – Context for header/footer

- Create a context for centrally controlling the header title as a string and the content of the footer as a react node.
- Set the header title for each task page to match the task name.

# Asynchronous operations

- Perform an action that will complete “later” E.g.: Fetch data, update server, delay something.
- React relies on state changes to know when to render something so we must translate async operations to state values.

- https://dummyjson.com

# Demo – List users

- Create a page that lists users.
- Make the list searchable and add a search box.
- Support deep-linking to a search result.
- Show user count in footer.

# Task – Posts by user

- Create a component that displays all posts for a given user.
- Provide user id from url.
- Update the users list so that you can click a user to display their posts.

# Task – Post page

- Create a page that displays a single post by id.
- Use the url to control which post should be displayed.
- Display all comments for the post.
- Allow sorting comments based on comment id or user name

# Demo – Create user

- Create a form where you can create a user by specifying a first name, last name and age.

# Task – Add comment

- Create components to allow users to add comments to a post.
- Submit comments to the dummy API.
- Invalidate the necessary comment cache upon submission.

# Task – Store

- Create a products page that lists products with images and some information.
- Add the ability to filter products by category.
- Create a single-product page that shows information about a single product.
- Allow users to click products on the product page to navigate to a single product.
- Bonus: Create a search field that allows users to search for posts by free text, if a category filter is set the search should be limited to that category.