PRACTICAL REACT

With TypeScript

Setup

https://github.com/rudfoss/practical-react

Follow instructions in README.md

bouvet

Our development environment

- Pnpm and Nx.dev powered repository configured as a monorepo
  - Project dependency tracking
  - Task orchestration
- Visual Studio Code some custom extensions and configurations
- Linting and formatting using BiomeJs
- React front-end bundled with Vite
  - Vite dev server with hot-reloading
  - Testing using Vitest and testing-library/react
- NestJs backend API
  - Auto-generated OpenAPI specification
  - Auto client-generation using NSwag (requires .NET 6+)

Agenda

1. What is React
2. Props, events and state
3. Lists and loops
4. Optimizing rendering
5. Styling
6. Testing
7. Organizing code
8. Routing
9. Services / Contexts
10. Server communication
11. Building the application (a bunch of tasks)

Jump to references 🚀

bouvet

What is React?

> A JavaScript library for building user interfaces

bouvet

Anatomy of React

The component

bouvet

Anatomy of React

```
function GreetPerson(props) {
  const userData = useContext(User)
  const [greeting, setGreeting] = useState("Hello there,")

  return (
    <div>
      <h1>{greeting} {userData.name}</h1>
      <input
        type="text"
        value={greeting}
        onChange={(event) => setGreeting(event.currentTarget.value)}
        disabled={props.disabled}
        />
    </div>
  )
}
```

Props

State

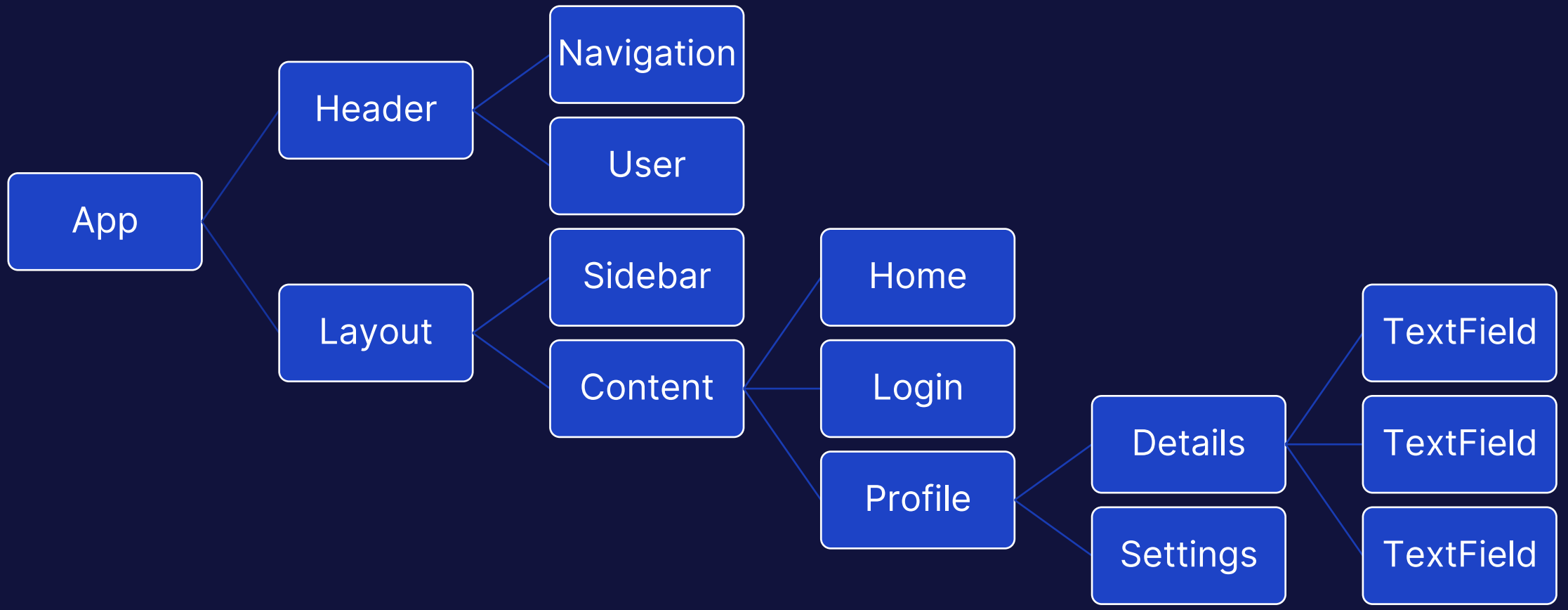
Context

JSX

bouvet

Anatomy of React

The component tree



bouvet

Props

- “Arguments” to a component.
- The component re-renders if the props change.
- Props are «packed» into an object and passed as the first argument to the component function.

```
<MyComponent foo="hey" bar={42} baz>Hello</MyComponent>
```

```
props = {
  foo: "Hey",
  bar: 42,
  baz: true,
  children: "Hello"
}
```

bouvet

Events

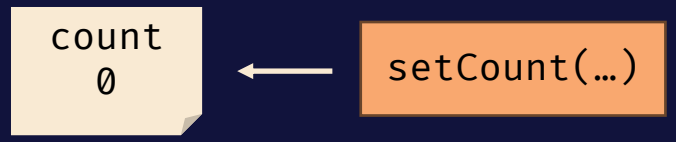
- Events are actions triggered by the user (or browser)
- Register a function to act on the event.

```
const onButtonClicked = (event: MouseEvent<HTMLButtonElement>) => {
  const buttonText = event.currentTarget.textContent
  console.log(`${buttonText} clicked`)
}

return <button onClick={onButtonClicked}>Click me</button>
```

State

- A “live” value the component can keep track of and update.
- “Setting” the value re-renders the component.

```
const [count, setCount] = useState(0)

return <button onClick={() => setCount(count + 1)}>{count}</button>
```



bouvet

<> Header

- Create a basic header with a <h1> title.
- Allow setting the header title from the “outside” using a “prop”.
- Add the component to the <App/>

<> TextField

- Create a component where the user can enter a single line of text.
- Add a “prop” to define the label of the field.
- Clicking the label should put focus in the text field (accessibility).
- Add “prop” to disable the field.

Hoisting state

- Lift the state “up” from a component to a parent.
- Allows sharing state between components.

<> CheckboxField

- Create a component where the use can check a box.
- Add a configurable label that focuses on the input when clicked.
- Add prop to disable the field.
- Hoist the “checked” state out of the CheckboxField so it can be reused.

<> Control header using text field

- Use the text entered in a text field to control the header text.
- Add a CheckboxField that controls whether the header shows the text from the TextField or the default “Hello World!”.

bouvet

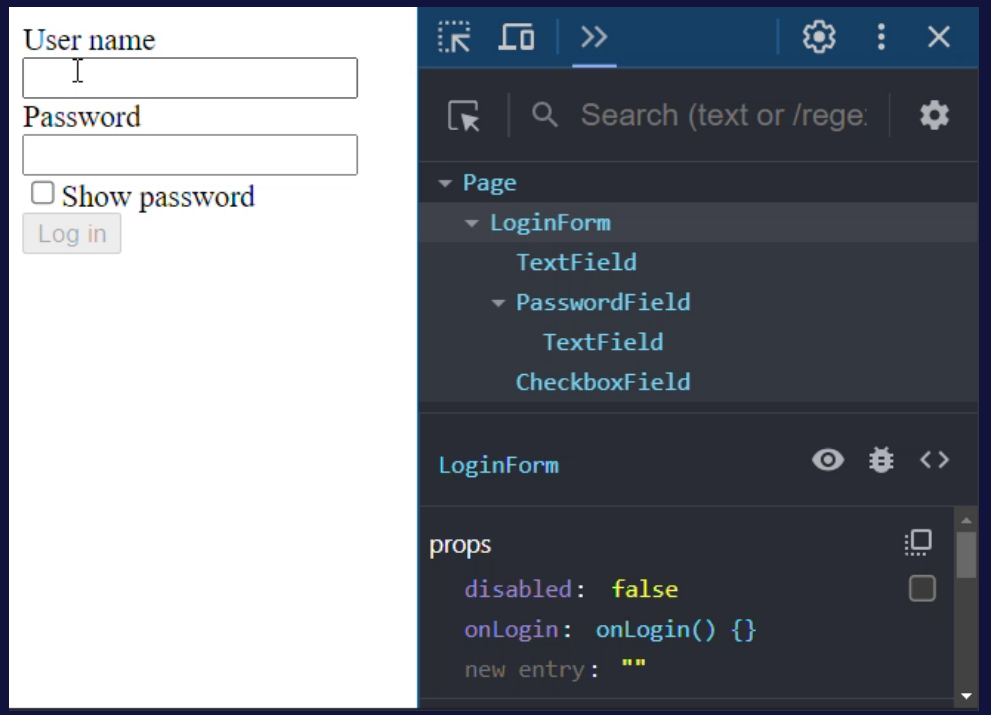
<> PasswordField

- Create a component where the user can enter a password.
- Add a configurable label that focuses on the input when clicked.
- Add optional props for:
  - making the password visible
  - disabling the field

bouvet

<> LoginForm

- User must enter a username and password
- Checkbox to show the password
- Login button (disabled until fields have values)
- Prop “onLogin” that receives the username and password
- Prop “disabled” that disables the entire form



Lists

- Render lists of items using the same components.
- Must keep track of which components corresponds to which item.
- Rules of keys.

<> GroupsTable

- Copy groups from the API and list them in a table with an “id” and “display name” columns.
- Add functions to sort groups by “id” or “display name” by clicking the column headings.
  - Bonus: Reverse sort order when clicking the same column heading multiple times

<> UsersTable

- Copy users from the API and list them in a table with “id” and “username” columns
- Add button on each row to “delete” the user.

Tests

- Unit tests using Vitest + testing-library/react

bouvet

<> GroupsTable.test

- Write some tests for the GroupsTable component
  - Test that it renders the same number of rows as there are groups
  - Test that the top group changes to the expected one

bouvet

Optimize rendering

- Make sure to only run code that needs to run
- Mark heavy code so that it only runs if values it depends on have changed (useMemo())
- Mark an entire component as “pure” so that it is only called if any of its props have changed (React.memo())
- Write immutable state changes
- Use dev tools to detect render problems

Immutability

- React works on the assumption that objects are immutable
- An immutable object cannot change it can only be replaced
- Optimizes for performance

Styling

- Style-prop on elements
- CSS files
- CSS modules
- CSS preprocessors
  - SASS, LESS, PostCSS
- CSS-in-js
  - styled-components, emotion, linaria, Griffel
  - Notes on using css-in-js

<> Style fields

- TextField/PasswordField
  - Move label above the input.
  - Add 16px top/bottom, 8px left/right margin around the entire field (label and input).
  - Use all available width for the input.
- CheckboxField
  - Add the same padding as the TextField.
  - Vertically center the checkbox and the label next to each other.
  - Add some spacing between the checkbox and the label.

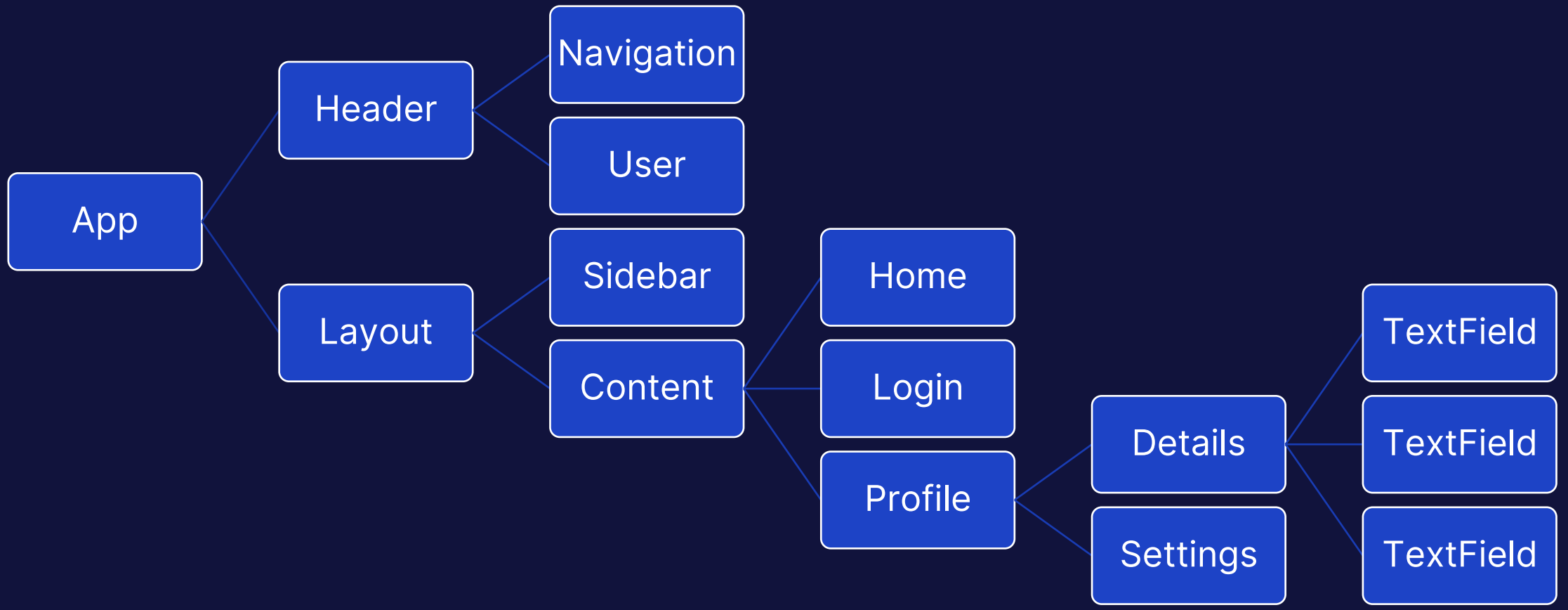
Organizing our code

- Apps vs libraries
  - An app can be built and deployed.
  - A library is used by one or more apps.
- Group libraries by domain
  - User components
  - Group components
  - Fields
  - UI
  - ++
- Utilize index files (barrel files) to control public vs private APIs.

Routing

- Selectively render pieces of our application tree based on the users current path.
- Extract parameters from the path and use them to select what content to render.
- Utilize History API in the browser.

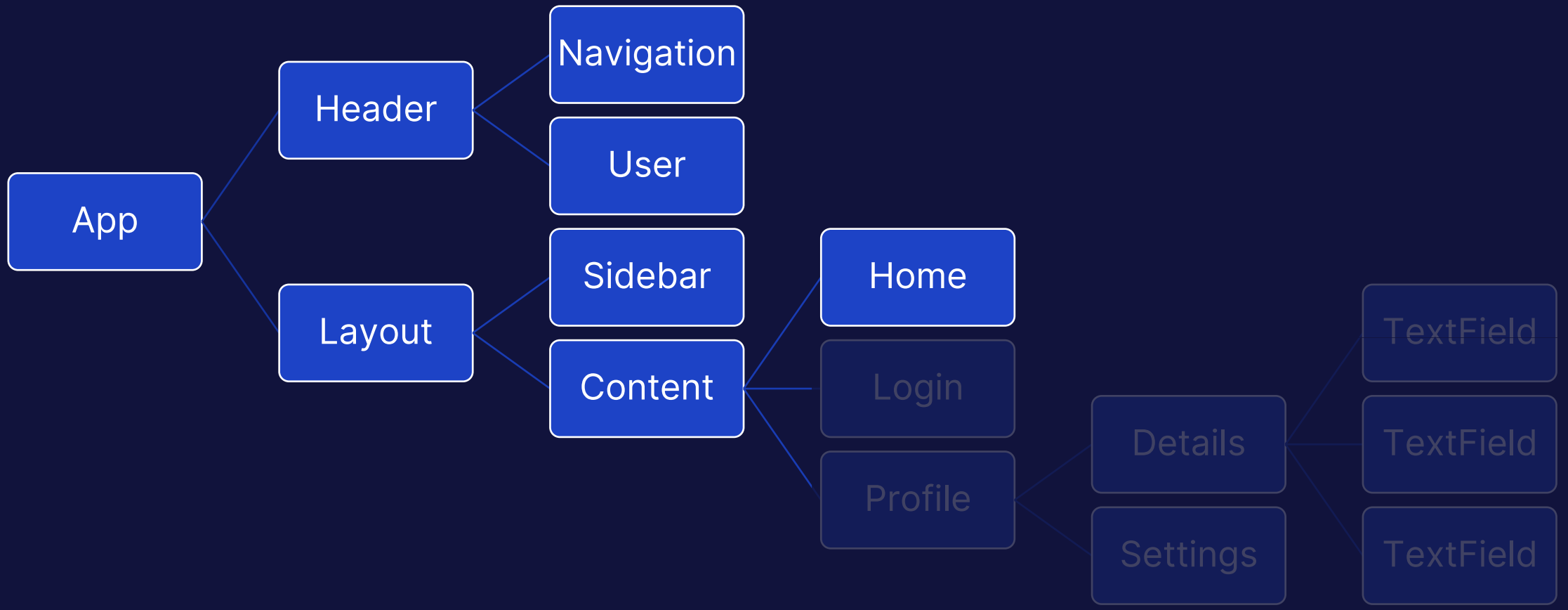
Routing



bouvet

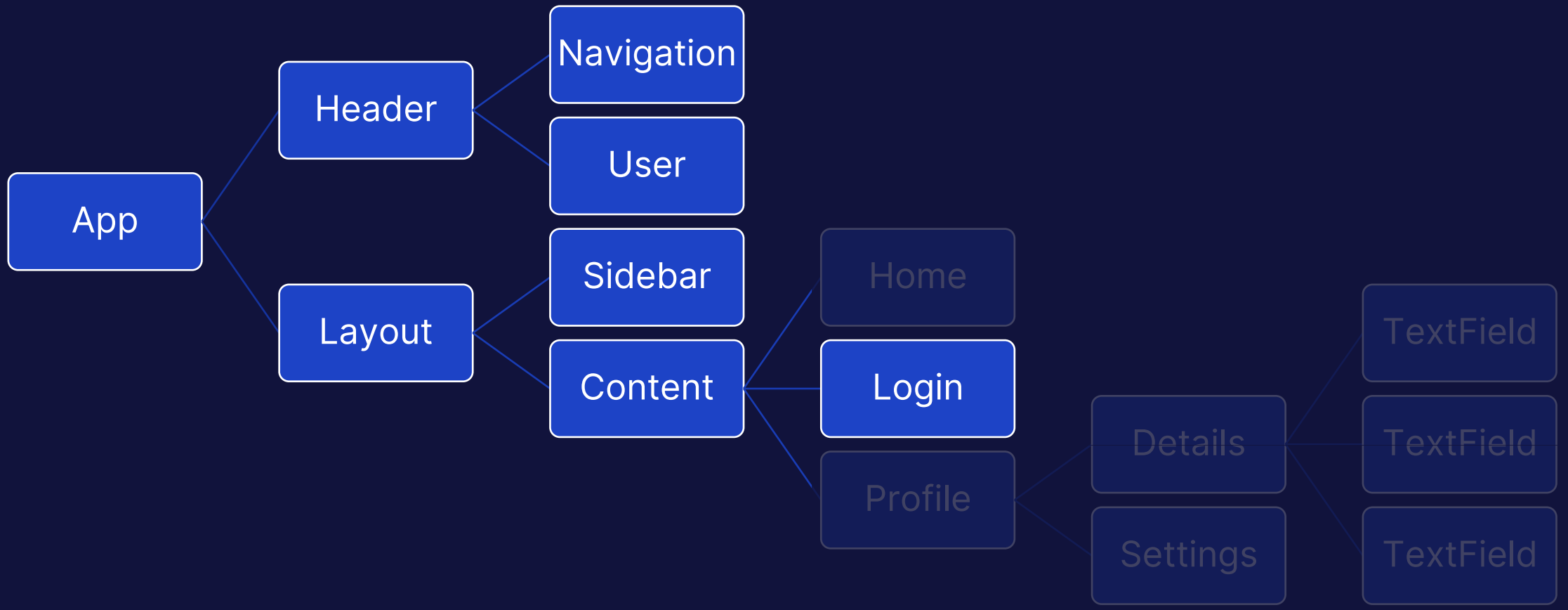
Routing

/


App
Header
Navigation
User
Layout
Sidebar
Content
Home
Login
Profile
Details
Settings
TextField
TextField
TextField

Routing

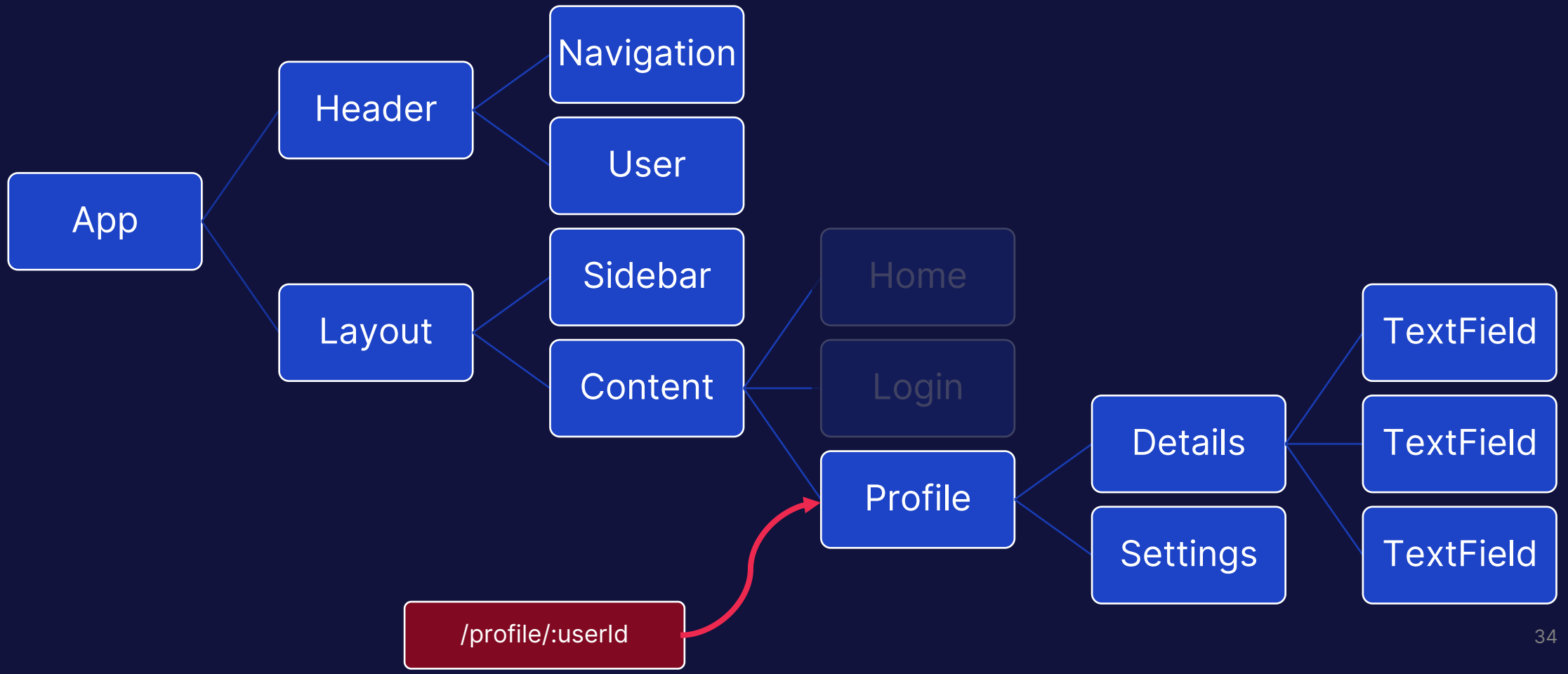
/login



bouvet

Routing with parameter

/profile/0e768eea-1c35-4024-8769-ef2dd62915e2



Setting up routing

- Define routes with pages
  - / -> Home Page
  - /users -> Users Page
- Provide routes
- Create error pages
- Navigation with links
- `pnpm add -w react-router-dom`

DisplayParameters

- Create a generic page that reads and displays parameters from the url pattern

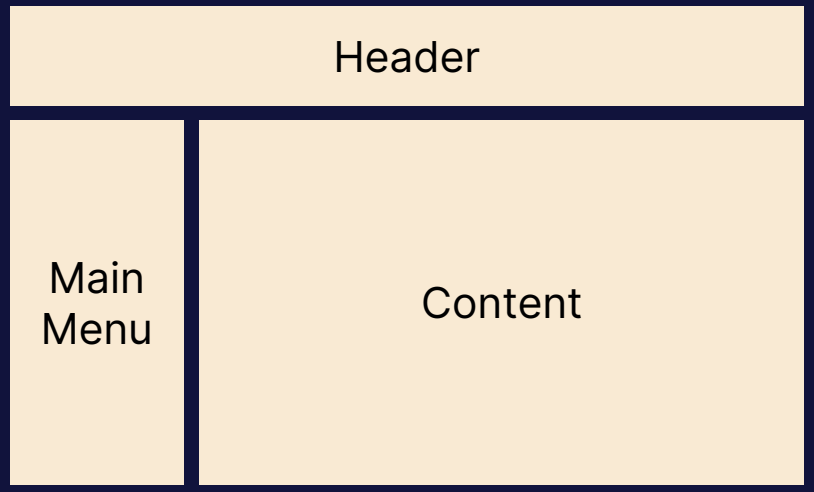
bouvet

Layouts

- Control the view surrounding the “route” component.

<> MainLayout

- Create a classic two-column layout with a header on top, navigation to the left and main content to the right.
- Use the layout for all pages in the application.



bouvet

<> GroupsPage

- Create a page that displays the static groups table
- Add a route for it at /groups
- Add a link to the groups page in the main menu
- Add another route that takes a group id as well: /groups/:id
- Highlight the group in the list if the parameter is present and the group exists

bouvet

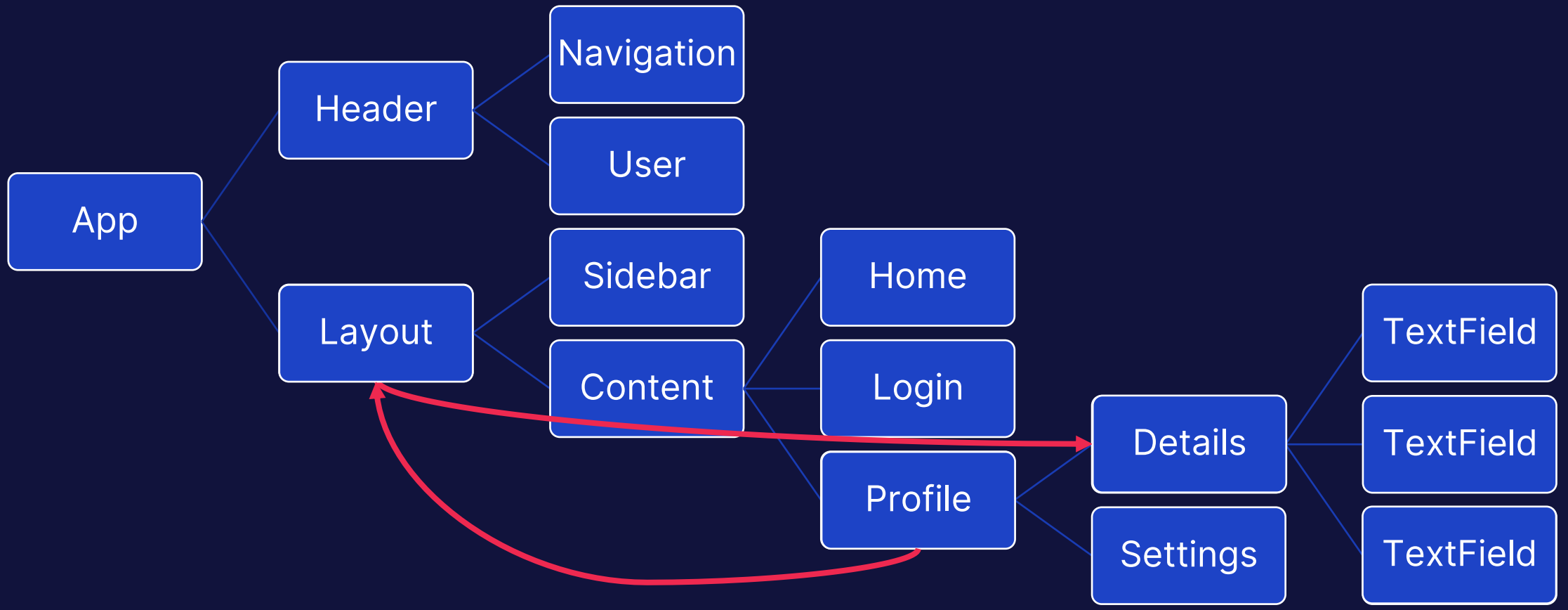
Code-splitting

- Divide our bundle into discreet chunks (usually based on routes)
- Reduce the size of the bundle that needs to be downloaded to start your app

Context

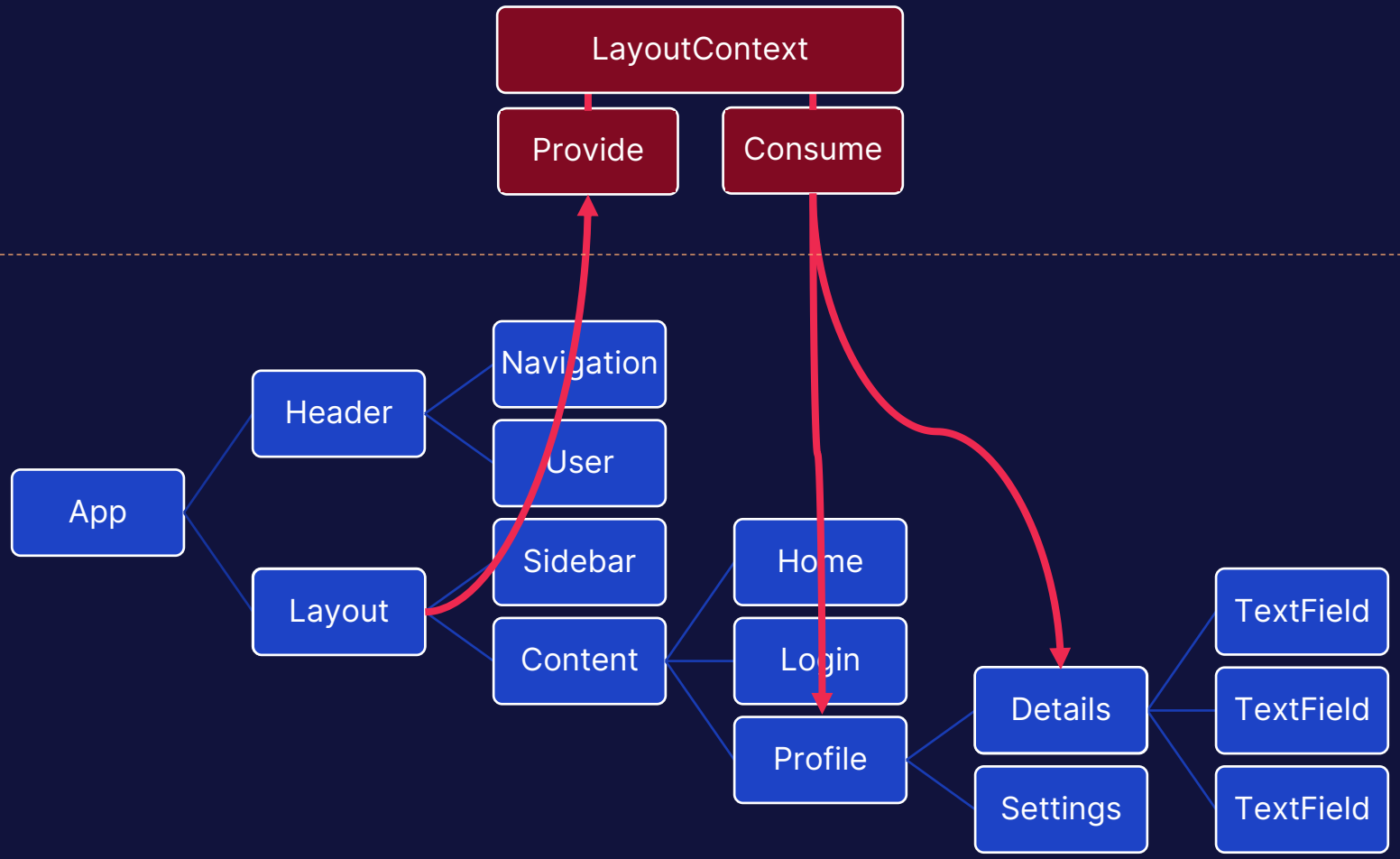
- Pass information between components that are not adjacent in the component tree.

Anatomy of Context



bouvet

Anatomy of Context



<> HeaderContext

- Create a mechanism so that pages can set the heading text.

Services using contexts

- **Provide** (inject) state or functionality to the application tree
- Manipulate shared state from nonadjacent components
- Provide a simple API that aids in proper usage

Server communication

- Get and set data on a remote server.
- Ensure the date on the client remains up to date.
- Show the user that data is being fetched and handle errors gracefully.
- Use types generate from the system that “owns” them.

Tanstack Query

- Client-side caching and invalidation
- Query state and error handling
- Retries and polling
- Reuse data across components
- TypeScript support ensures type-safe client code
- `pnpm add -w @tanstack/react-query` (link)

Tanstack Query

Two types of requests

Query

- Get data from a server with optional parameters.
- Runs as soon as a useQuery() hook is encountered.
- Read-only.
- Usually corresponds to HTTP verbs: GET.

Mutation

- Change data on the server with optional parameters.
- useMutation() hook runs only when explicitly requested.
- May invalidate locally cached data.
- Usually corresponds to HTTP verbs: POST, PUT, PATCH, DELETE.

<> ApiStats

- Create a component that displays statistics from the API
- Create a page and route for it
- Show a loading spinner while it loads

bouvet

<> ApiHealth

- Create a component that shows health information from the API
- Create a page and route for it
- Show a loading spinner while it loads

Basic login

- Use the login form to log a user in and display the result.
- Disable the form while the user is being logged in.
- Implement a logout button that logs the user out. It should only be visible IF the user is previously logged in.
- Add a loading indicator that can be displayed while the login is in-flight.

Building the application

- What remains now is using what we have learned to build out the application.
- We will most likely not be able to complete all these, but you can find example solutions in the solution/* branches.

References

https://react.dev
React documentation

https://nx.dev/
The tool used to generate code and manage the monorepo we are working in

https://vitejs.dev/
Compiler and bundler for our front-end code.

https://reactrouter.com/
A robust router for react

https://griffel.js.org
Atomic-CSS-in-JS library with ahead-of-time compilation

https://nextjs.org
“The” React framework

https://tanstack.com/query
Remote-state cache and orchestration library

- https://tkdodo.eu/blog/practical-react-query
  A comprehensive blog post about most of the features of Tanstack Query
- https://query.gg
  The official Tanstack Query course

https://github.com/pmndrs/zustand
Simple global state manager

https://biomejs.dev
An opinionated code linter and formatter

https://eslint.org
Linting tool for enforcing coding standards

- https://github.com/sindresorhus/eslint-plugin-unicorn
  Extra linting rules

https://playwright.dev/
End-to-end testing using multiple browsers

https://github.com/RicoSuter/NSwag
Tool for generating TypeScript (and other) clients from OpenApi

https://www.mockaroo.com
Tool for generating test data

https://editorconfig.org/
IDE-agnostic formatting and style configurations

https://monorepo.tools/
An overview of what a monorepo is

MDN Web Docs (mozilla.org)
Reference documentation and tutorials for HTML, CSS, JavaScript and more

ImmerJs
Helps writing immutable objects