PRACTICAL REACT WITH TYPESCRIPT

Setup

-
  -
  -
  -
-
  - `git clone https://github.com/rudfoss/practical-react-with-typescript.git`

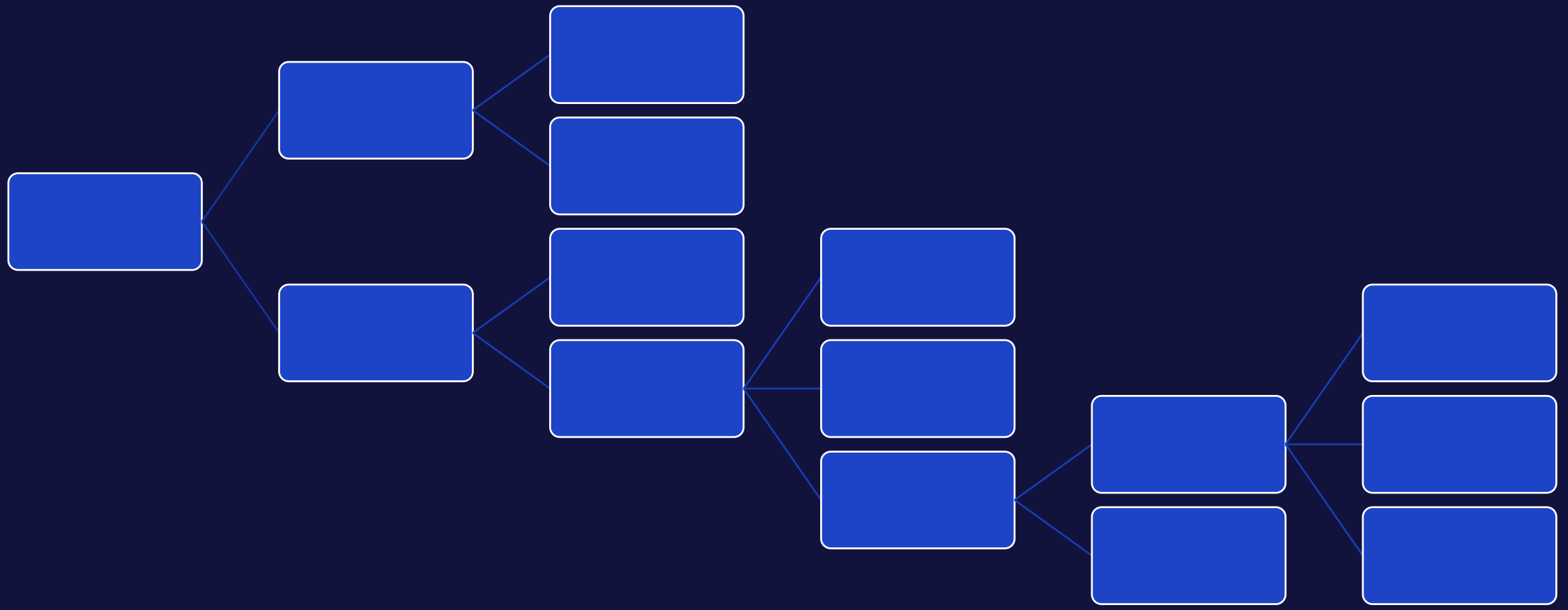
bouvet

Agenda

-
  -
  -
  -
  -
-
  -
  -
  -
  -
  -
-
  -
  -
  -
  -
  -
  -
  -
  -

bouvet

Anatomy of React



bouvet

TextField

-
-
-
-

bouvet

<> BooleanField

-
-
-

bouvet

Reusable fields

-

Anatomy of a component

```tsx
export interface TextFieldProps {
  label: string
}
```

props

component

```tsx
export const TextField = ({ label }: TextFieldProps) => {
  const id = useId()
  const [value, setValue] = useState("")

  return (
    <div>
      <label htmlFor={id}>{label}</label>
      <input id={id} type="text" value={value} onChange={(evt) => setValue(evt.target.value)} />
      <p>{value}</p>
    </div>
  )
}
```

bouvet

```tsx
export interface TextFieldProps {
  label: string
}

export const TextField = ({ label }: TextFieldProps) => {
  const id = useId()
  const [value, setValue] = useState("")

  return (
    <div>
      <label
```

Anatomy of a component

use* hooks

```tsx
export interface TextFieldProps {
  label: string
}

export const TextField = ({ label }: TextFieldProps) => {
  const id = useId()
  const [value, setValue] = useState("")

  return (
    <div>
      <label htmlFor={id}>{label}</label>
      <input id={id} type="text" value={value} onChange={(evt) => setValue(evt.target.value)} />
      <p>{value}</p>
    </div>
  )
}
```

bouvet

Anatomy of a component

useState

```tsx
export interface TextFieldProps {
  label: string
}

ex                                    dProps) => {
  const id = useId()
  const [value, setValue] = useState("")

  return (
    <div>
      <label htmlFor={id}>{label}</label>
      <input id={id} type="text" value={value} onChange={(evt) => setValue(evt.target.value)} />
      <p>{value}</p>
    </div>
  )
}
```

Anatomy of a component

```tsx
export interface TextFieldProps {
  label: string
}

export const TextField = ({ label }: TextFieldProps) => {
  return (
    <div>
      <label htmlFor={id}>{label}</label>
      <input id={id} type="text" value={value} onChange={(evt) => setValue(evt.target.value)} />
      <p>{value}</p>
    </div>
  )
}
```

jsx

bouvet

Anatomy of a component

```tsx
export interface TextFieldProps {
  label: string
}

export const TextField = ({ label }: TextFieldProps) => {
  const id = useId()
  const [value, setValue] = useState("")

  retur
    <div>
      <label htmlFor={id}>{label}</label>
      <input id={id} type="text" value={value} onChange={(evt) => setValue(evt.target.value)} />
      <p>{value}</p>
    </div>
  )
}
```

children label

children label TextField

bouvet

Anatomy of a component

```tsx
export interface TextFieldProps {
  label: string
}

exp
  const id = useId()
  const [value, setValue] = useState("")

  return (
    <div>
      <label htmlFor={id}>{label}</label>
      <input id={id} type="text" value={value} onChange={(evt) => setValue(evt.target.value)} />
      <p>{value}</p>
    </div>
  )
}
```

useState

destructure

Anatomy of a component

```tsx
export interface TextFieldProps {
  label: string
}

export const TextField = ({ label }: TextFieldProps) => {
  const id = useId()
  const [value, setValue] = useState("")

  return (
    <div>
      <label htmlFor={id}>{label}</label>
      <input id={id} type="text" value={value} onChange={(evt) => setValue(evt.target.value)} />
      <p>{value}</p>
    </div>
  )
}
```

value input

onChange function

bouvet

Anatomy of a component

- Component:

- Props
- Hooks
- State
- Children
- JSX
- { }

JSX

Anatomy of an event

```tsx
export interface TextFieldProps {
  label: string
}

export const TextField = ({ label }: TextFieldProps) => {
  const id = useId()
  const [value, setValue] = useState("")

  return (
    <div>
      <label htmlFor={id}>{label}</label>
      <input id={id} type="text" value={value} onChange={(evt) => setValue(evt.target.value)} />
      <p>{value}</p>
    </div>
  )
}
```

input*

input **onChange** **input**

bouvet

Anatomy of an event

```tsx
export interface TextFieldProps {
  label: string
}

export const TextField = ({ label }: TextFieldProps) => {
  const id = useId()
  const [value, setValue] = useState("")

  return (
    <div>
      <label htmlFor={id}>{label}</label>
      <input id={id} type="text" value={value} onChange={(evt) => setValue(evt.target.value)} />
      <p>{value}</p>
    </div>
  )
}
```

event handler

setter

bouvet

Anatomy of an event

```tsx
export interface TextFieldProps {
  label: string
}
```

State change → re-render updated data

```tsx
export const TextField = ({ label }: TextFieldProps) => {
  const id = useId()
  const [value, setValue] = useState("")

  return (
    <div>
      <label htmlFor={id}>{label}</label>
      <input id={id} type="text" value={value} onChange={(evt) => setValue(evt.target.value)} />
      <p>{value}</p>
    </div>
  )
}
```

Anatomy of an event

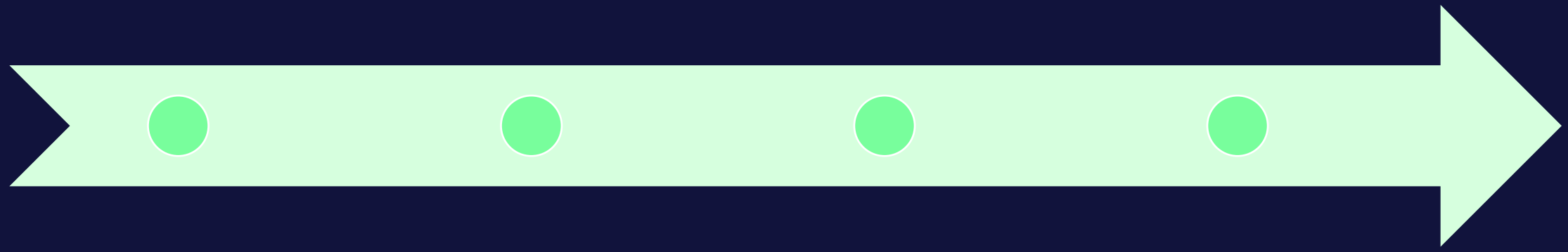
```tsx
export interface TextFieldProps {
  label: string
}

export const TextField = ({ label }: TextFieldProps) => {
  const id = useId()
  const [value, setValue] = useState("")

  return (
    <div>
      <label htmlFor={id}>{label}</label>
      <input id={id} type="text" value={value} onChange={(evt) => setValue(evt.target.value)} />
      <p>{value}</p>
    </div>
  )
}
```

value

updating the UI

bouvet

Anatomy of an event



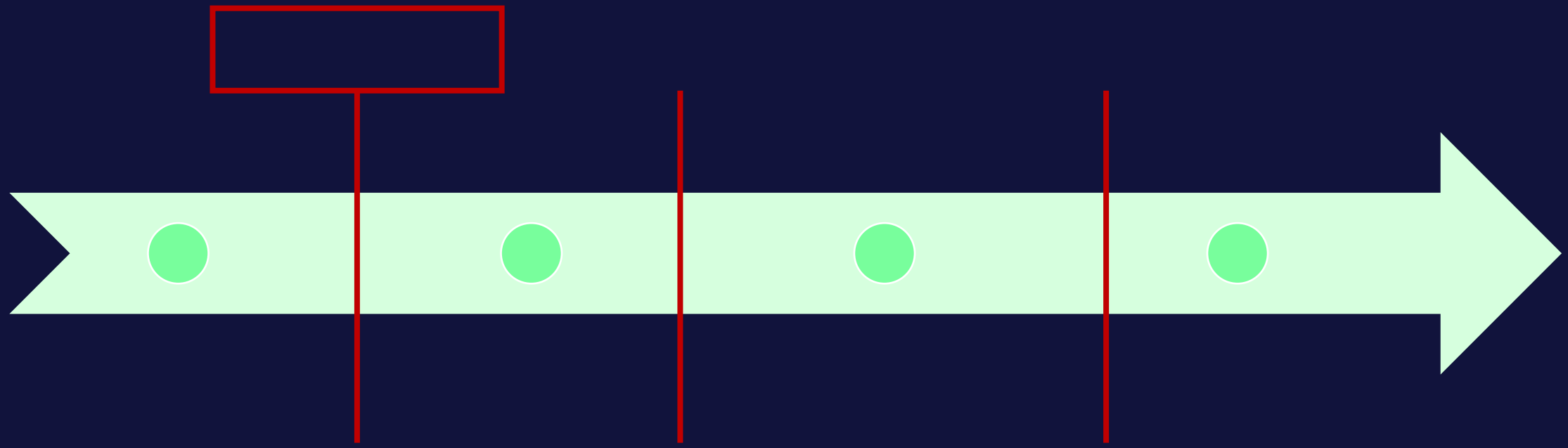
bouvet

Component lifecycle

-
  -
-
  -
-
  -

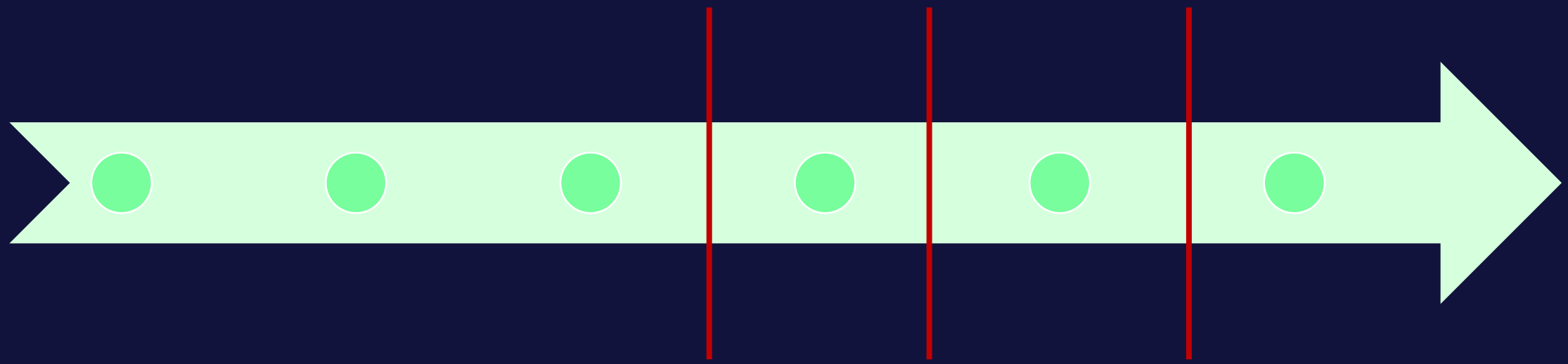
Component lifecycle

TextField



Component lifecycle

TextField



bouvet

ClickUntil

-
-
-
-
-

bouvet

<> NumericField

-
-
-
  -
  -
  -
-

bouvet

<> ClickUntilForm

-

-

bouvet

Styling

-
  -
  -
  -

bouvet

Style TextField

bouvet

<> Style components

-
-
-

bouvet

EoD 1

bouvet

Organizing our repository

-
  -
  -
-
  -
  -
  -
  -
  -
  -

bouvet

Immutability

-
-

bouvet

UserDetails

-
-
  -
  -
  -
  -
  -
  -
-
-

bouvet

<> GroupDetails

-
-
-

bouvet

Loops

-
-

bouvet

useMemo

-

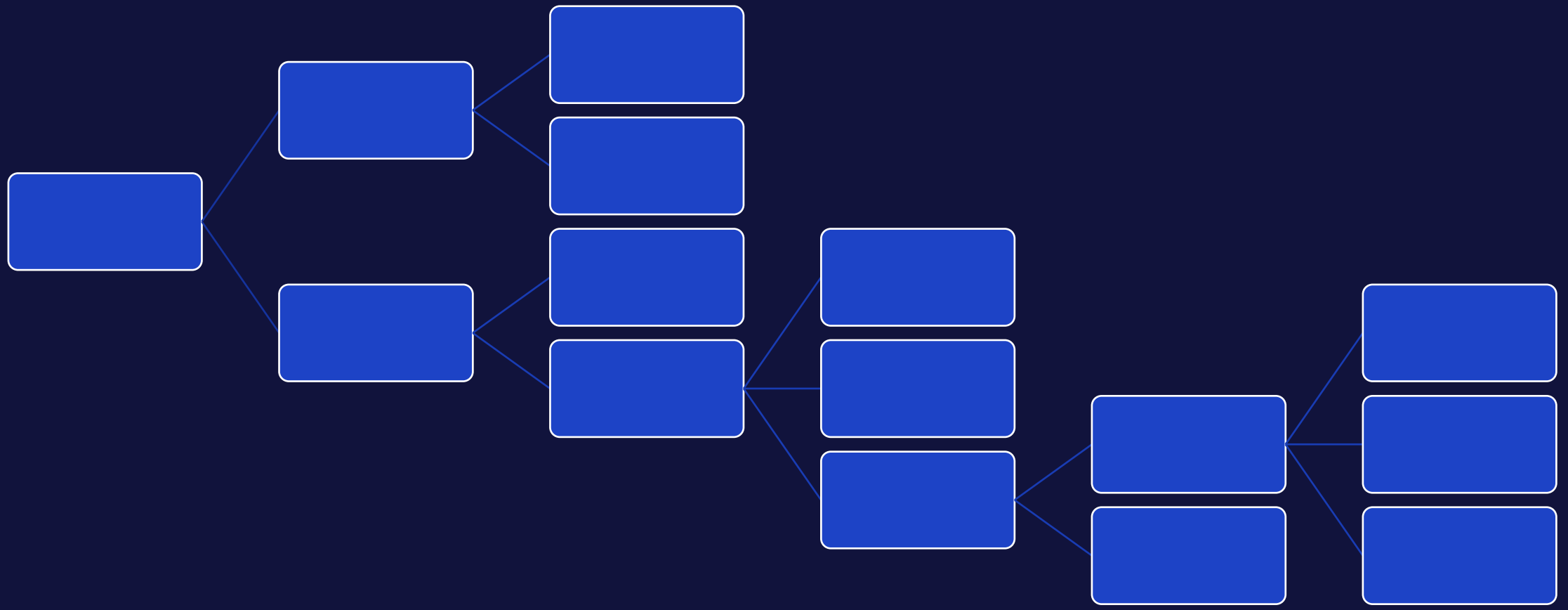
bouvet

Routing

-
-

bouvet

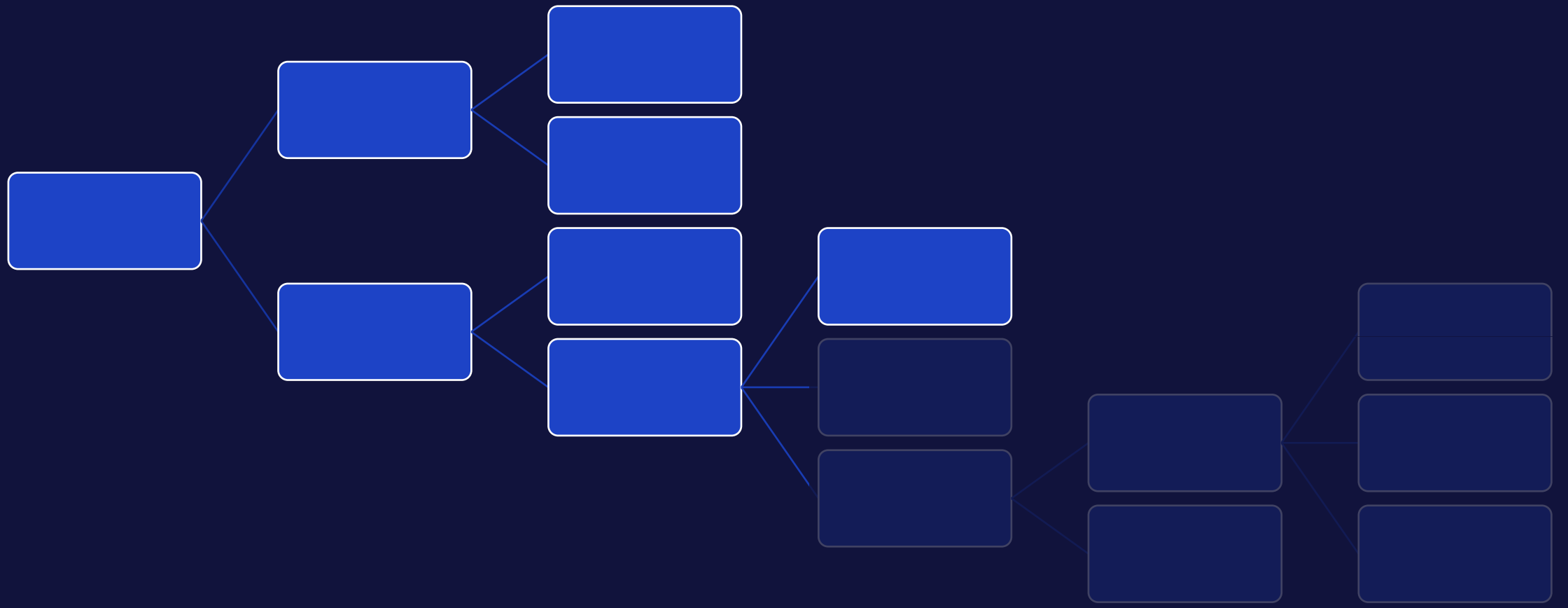
Anatomy of routing



bouvet

Anatomy of routing

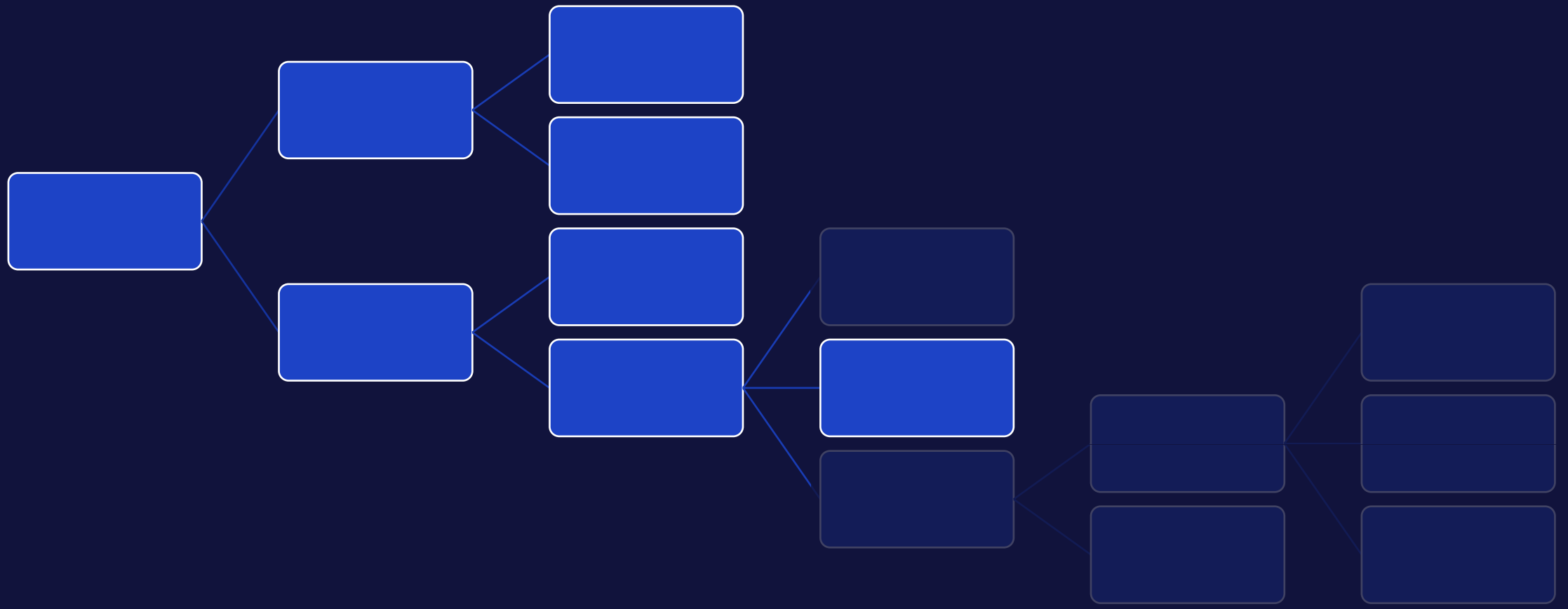
/home



bouvet

Anatomy of routing

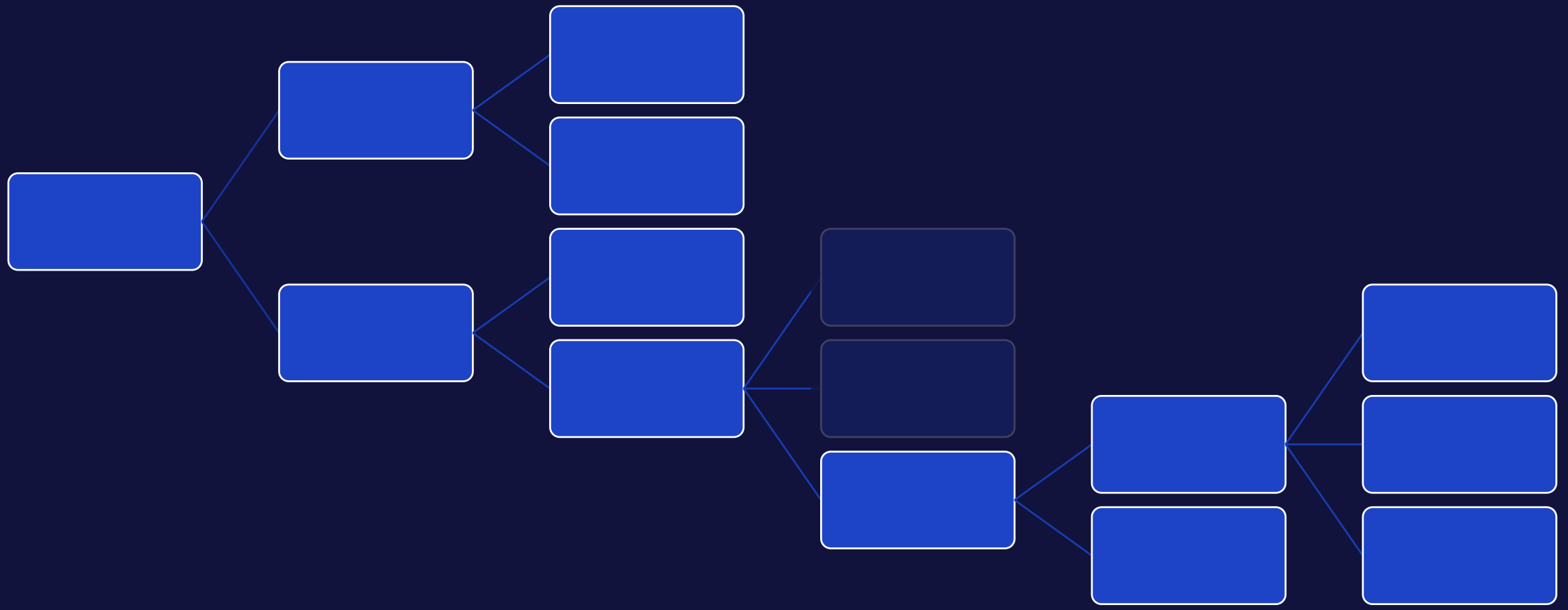
/login



bouvet

Anatomy of routing

/profile



bouvet

HomePage and UserDetailsPage

-
-
-
-

bouvet

MainLayout

-
-

bouvet

<> UsersTable Page and UserDetails Page

-
-
-
-
-

/users

bouvet

EoD 2

bouvet

Hooks

-
-
  -

bouvet

Most common react hooks

- useId:
- useState:
- useMemo:
- useEffect:

- useRef: does not

useRandom

-

<> useWindowTitle

-

-

bouvet

Error handling

-
- `errorElement`

bouvet

<> useThrowIfNetworkUnavailable

-
-

bouvet

TypeScript Generics

•

bouvet

TypeScript Generics

```ts
interface ContainerFor<TType> {
  id: string
  value: TType
}
```

```ts
const stringContainer: ContainerFor<string> = {
  id: "123",
  value: "foo"
}

const booleanContainer: ContainerFor<boolean> = {
  id: "123",
  value: true
}

interface Name {
  firstName: string
  lastName: string
}
const objectContainer: ContainerFor<Name> = {
  id: "123123",
  value: {
    firstName: "Test",
    lastName: "Testington"
  }
}
```

TypeScript Generics

```ts
interface ContainerFor<TType extends string | boolean> {
  id: string
  value: TType
}
```

```ts
const stringContainer: ContainerFor<string> = {
  id: "123",
  value: "foo"
}

const booleanContainer: ContainerFor<boolean> = {
  id: "123",
  value: true
}
```

TypeScript Generics

```ts
interface GenericInterface<T> {
  id: string
  value: T
}

type GenericType<T> = {
  id: string
  value: T
}

const genericFn = <T, T2>(firstArg: T, secondArg: T2) => {
  //...
}

function genericFn2<T, TReturn>(firstArg: T): TReturn {
  //...
}
```

TypeScript Generics with React

```tsx
export interface GenericComponentProps<TType> {
  label: string
  value: TType
}

export const GenericComponent = <TType,>(props: GenericComponentProps<TType>) => {
  // ..
}
```

bouvet

<> ChoiceField

-
-
-
-

bouvet

<> ChoiceField

-
-
-
-
-

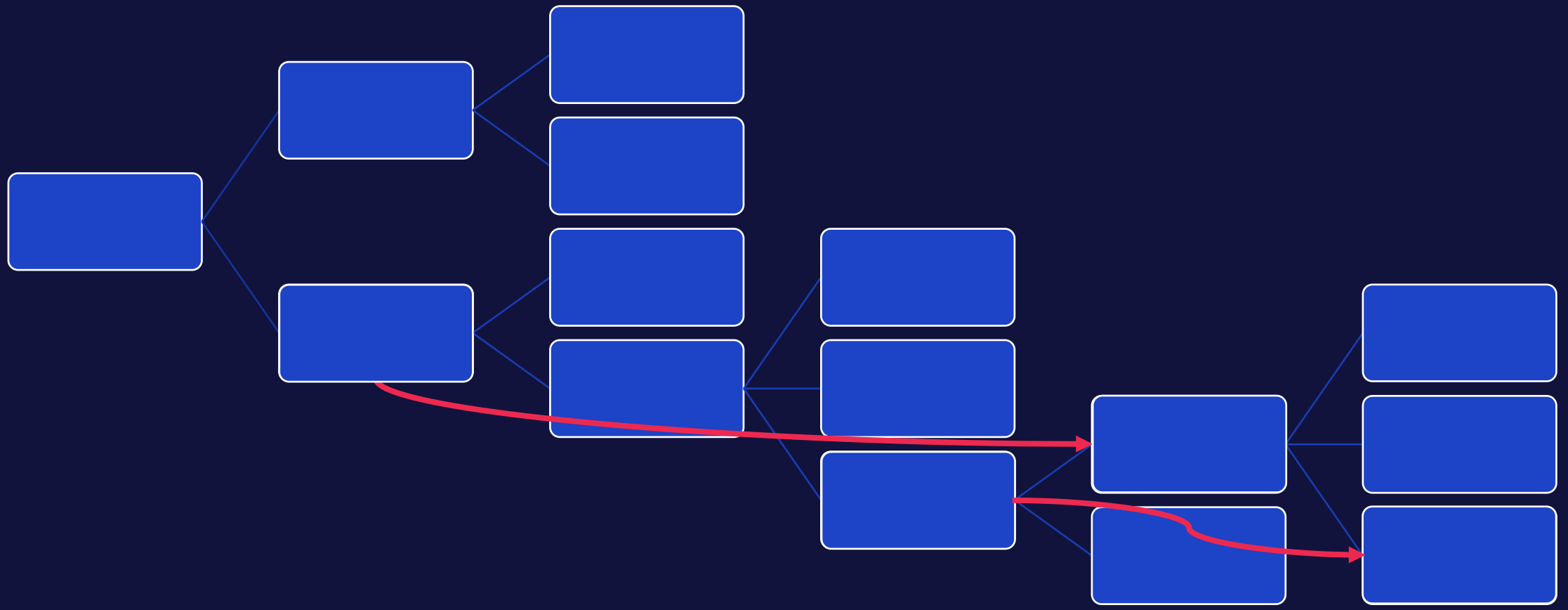
bouvet

Contexts

-
-

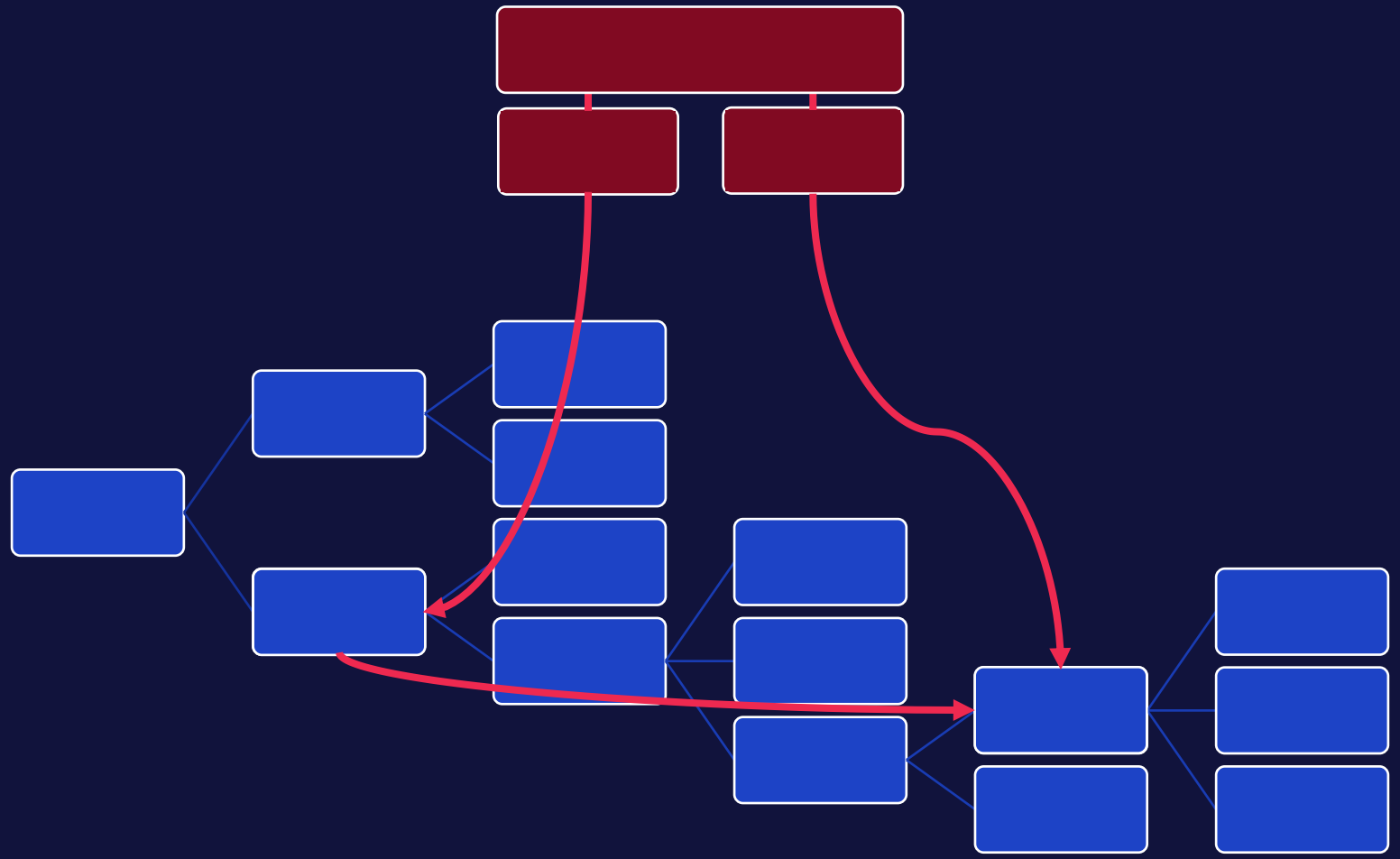
bouvet

Anatomy of Context



bouvet

Anatomy of Context



bouvet

NavContext

bouvet

<> DisableFieldsContext

-
-
-
-

bouvet

State management scopes

-
-
-
-

bouvet

Communication with a server

-
-
-

bouvet

Generating clients

-
-
-

bouvet

<> ServerSideUserTable

-
-
-
-

bouvet

UserDetails

-
-
-
-

bouvet

Testing

-
-

bouvet

Optimizations

- **Avoid unecessary re-renders**
  -
  -
-
  -
-
  -
-
  -

Resources

bouvet