

UI/UX ENTWICKLUNG MIT

REACTJS

**METHODEN
& KONZEPTE**



PRAXIS

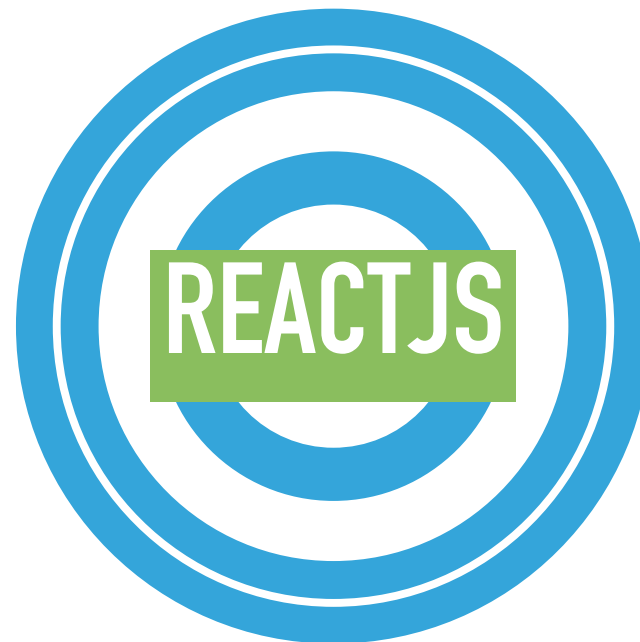


STRUKTUR

KOMPLEXITÄT

WIEDERVERWENDUNG

SOLL/IST



**INFORMATIONEN
ARCHITEKTUR**

**DOMAIN
DRIVEN
DESIGN**

**ATOMIC
DESIGN**

TESTING

ZUSAMMENFASSUNG

- ▶ Interdisziplinäre Zusammenarbeit
- ▶ Verbinden von Nutzererwartung und fachlich/technischer Lösung
- ▶ Aufbau und Festigung von explizitem Wissen
- ▶ Prägen möglichst exakter Begriffe und deren Bedeutung
- ▶ Festlegen von Kontext und Struktur durch Abgrenzung und Gruppierung
- ▶ Organisation von Abhängigkeiten
- ▶ Sicherung der Integrität zwischen Anwendungskomponenten

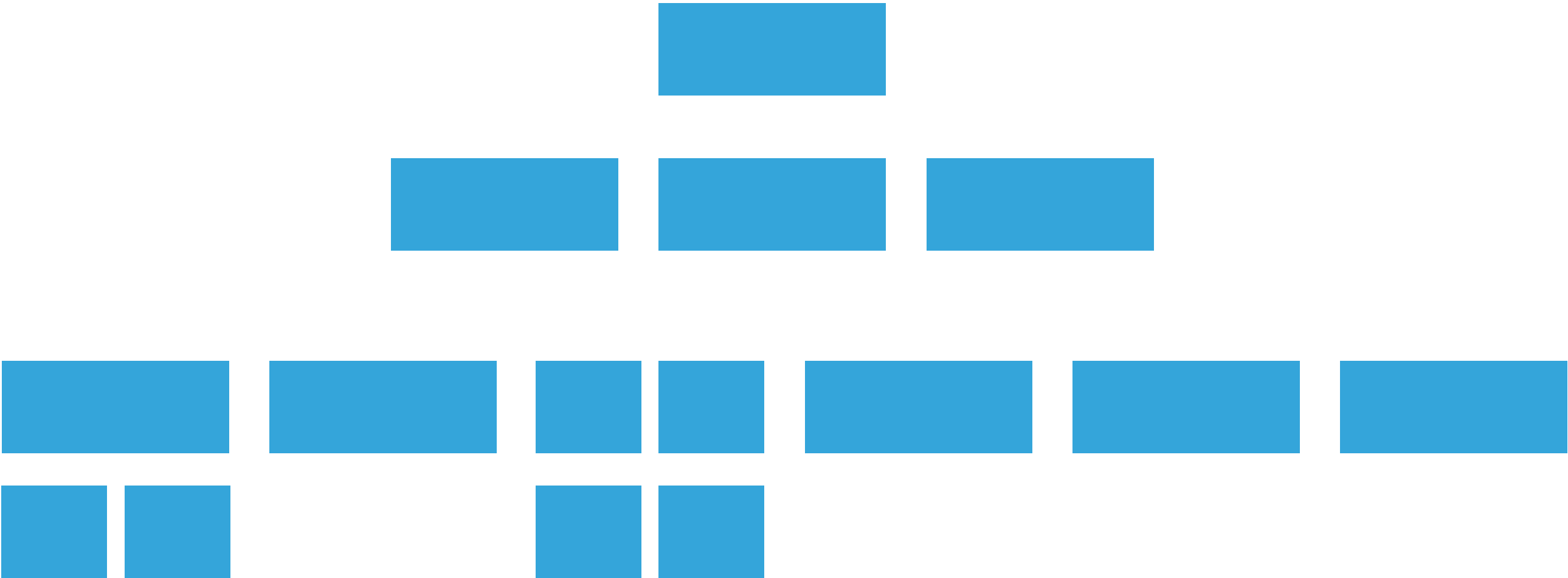
INFORMATION ARCHITECTURE

WAS IST EIGENTLICH INFORMATIONSSARCHITEKTUR?

EINE EFFEKTIVE IA WAR UND IST ALSO EINE STRUKTUR, DIE SICH DEM NUTZER NICHT IN DEN WEG STELLT, SONDERN HILFT INNERHALB EINES (MEIST DIGITALEN) INFORMATIONSSYSTEMS EIN GEWÜNSCHTES ZIEL ZU ERREICHEN.

Jan Jursa

STRUKTUR



WAS BRAUCHT DER BENUTZER?

- ▶ **Organization**

Gruppierung von Informationen nach z.B. Zeit, Thema, Funktion, Aufgabe, Prozess, Publikum, Attribut, Facette, Standort

- ▶ **Benennung**

Erarbeiten von Begriffen und Gruppierungen

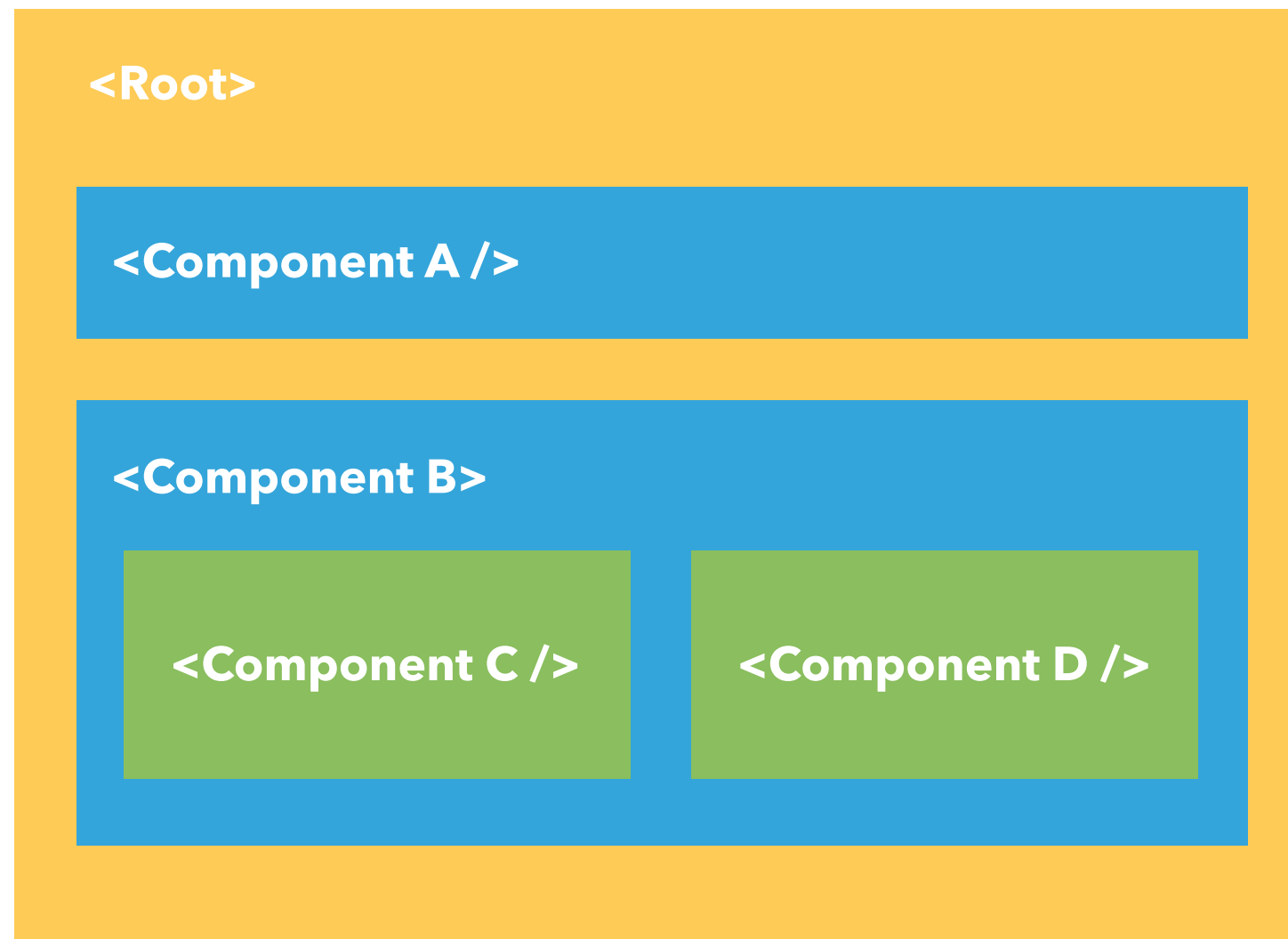
- ▶ **Priorisieren**

Anordnen der Inhalte

- ▶ **Verbindungen**

Verknüpfen von Aufgaben und Lösung

REACTJS HIERARCHIE



REACTJS KONZEPTE UND ARCHITEKTUR

- ▶ DOM vs. VDOM
- ▶ Redering
- ▶ Eltern-Kind Strukturen
- ▶ Unidirektionale Datenbindungen
- ▶ Ereignisse

REACTJS UNIDIREKTIONALE DATENBINDUNG

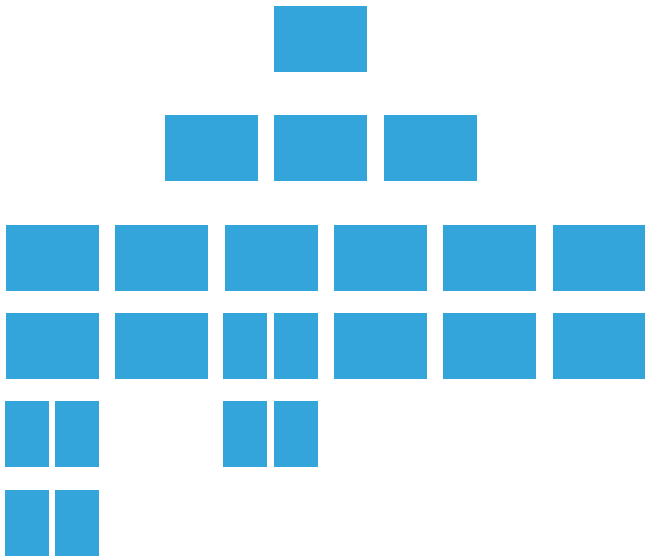
DATEN



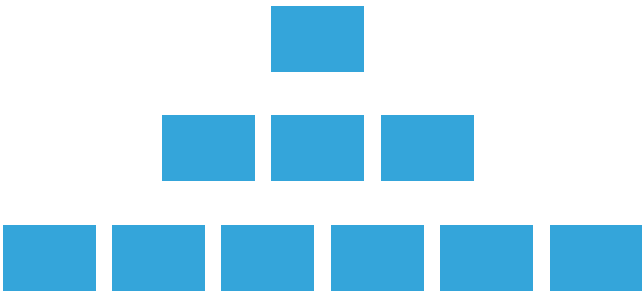
EREIGNISSE

REACTJS ARCHITEKTUR

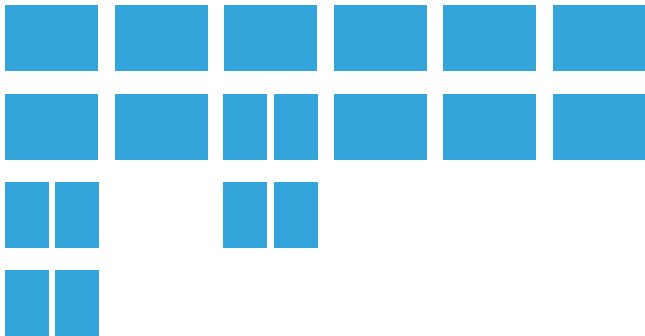
APP



APP A



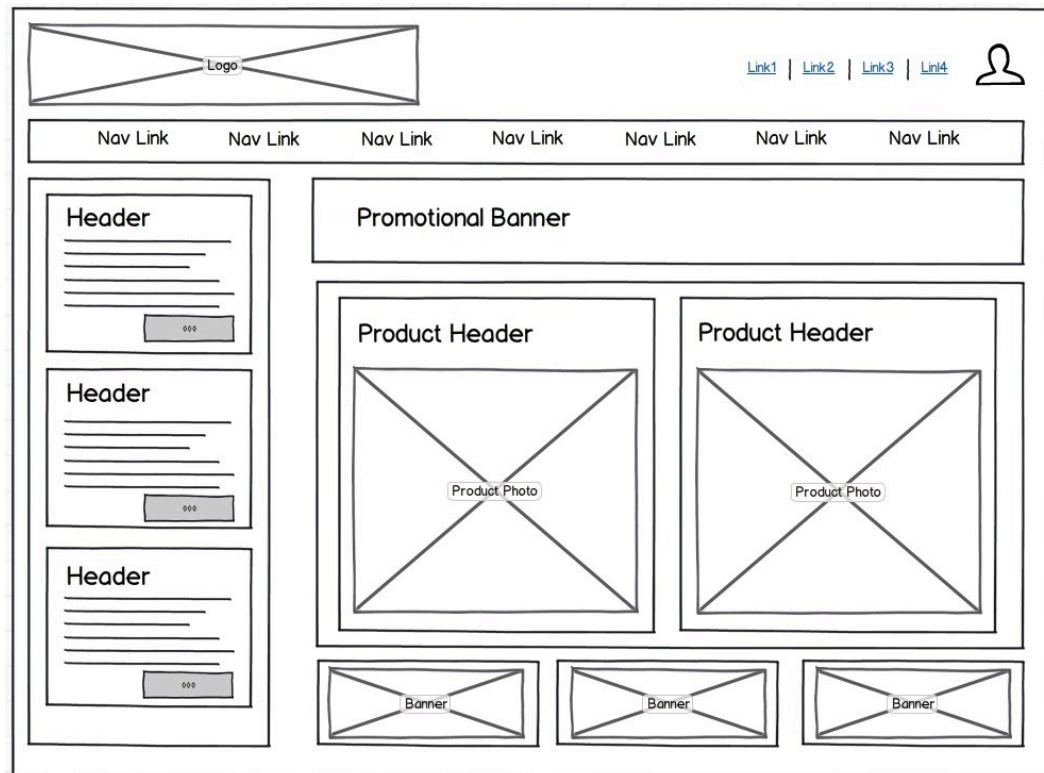
SHARED



TOOLS

- ▶ Storyboard & Wireframes - **Balsamiq**
- ▶ Projekte/Workspaces - **Yarn / Lerna**
- ▶ App/Pages/Component Hierarchie - **React Router**

STORYBOARD & WIREFRAMES



- ▶ Benutzerführung - Gruppierung, Benennung und Interaktion
- ▶ Verknüpfung von Daten und Aufgaben
- ▶ Navigation und UI Mockups

PROJEKTE/WORKSPACES

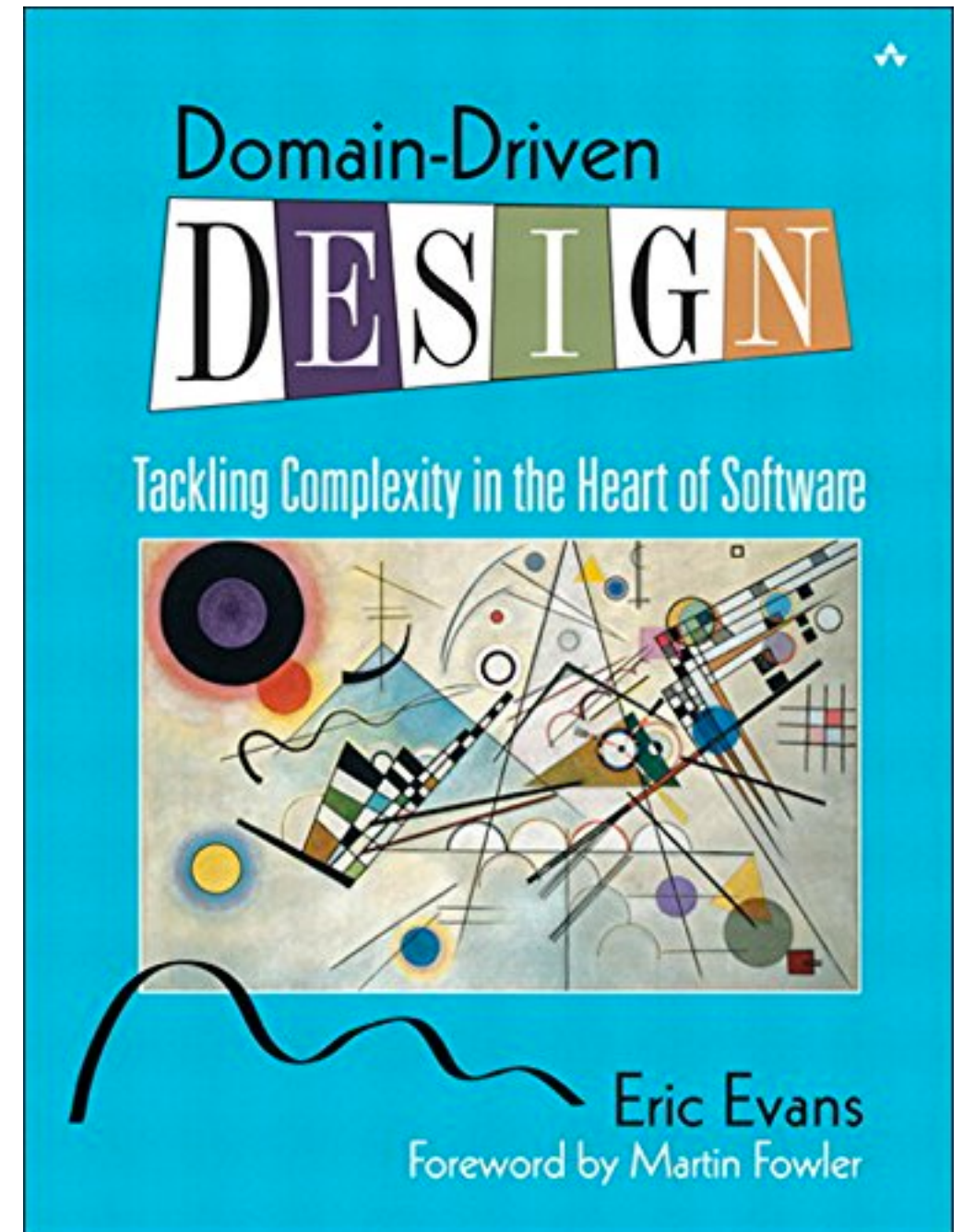
- ▶ Isolieren und Gruppieren von fachlichen und technischen Arbeitsbereichen
- ▶ Verwaltung von Abhängigkeiten
- ▶ Explizite APIs, Versionierung und Dokumentation

ZUSAMMENFASSUNG

- ▶ Fachliche Begriffe und Zusammenhänge erkennen, prägen, anordnen und verknüpfen
- ▶ Gewährleisten einer hohen Informationsintegrität
- ▶ Strukturierung von komplexen Informationen, Aufgaben und deren Beziehungen in ein Wireframes/Storyboard
- ▶ Erkennen, isolieren, benennen und etablieren von UI/UX Mustern
- ▶ Designgrundlage für ReactJS Komponenten-APIs, Datenbindung und Ereignisse

FRAGEN?

DOMAIN DRIVEN DESIGN

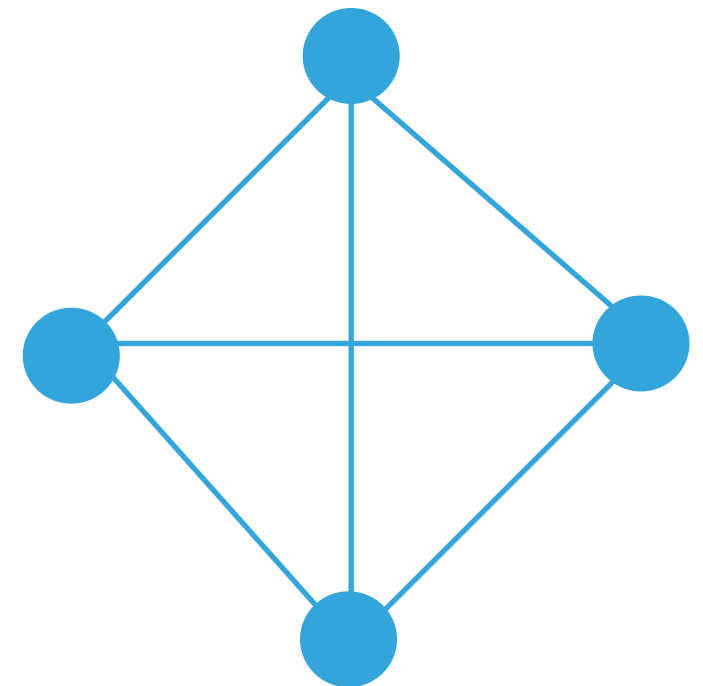
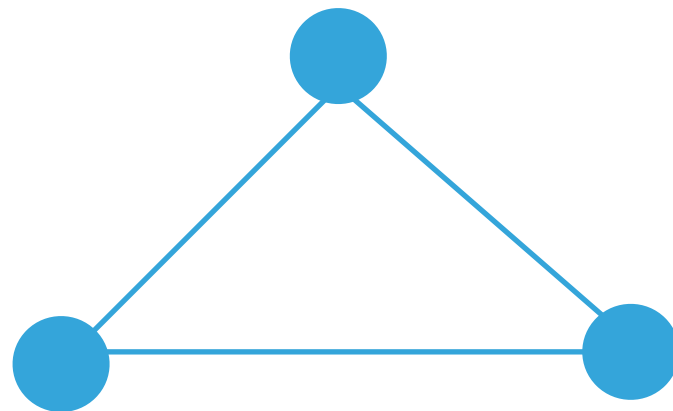
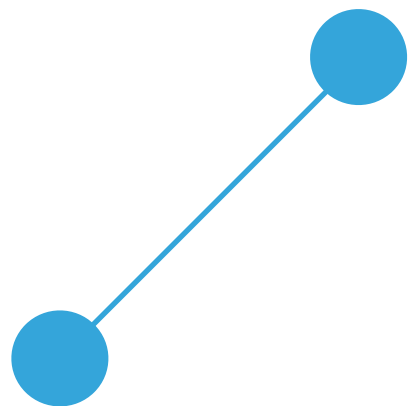


WAS IST EIGENTLICH DDD?

DOMAIN-DRIVEN DESIGN, KURZ DDD, BESCHREIBT VORGEHENSWEISEN, DIE KOMPLEXE SOFTWARE-PROJEKTE TRANSPARENTER FÜR ALLE BETEILIGTEN MACHEN SOLLEN. GLEICHZEITIG DEFINIERT ES EINE REIHE VON TECHNIKEN UND ELEMENTEN, MIT DENEN EIN OPTIMIERTES DOMÄNENMODELL ERREICHT WERDEN SOLL.

Eric Evans

KOMPLEXITÄT



...KOMMUNIKATION, ABHÄNGIGKEITEN, ZUSTÄNDE, KOMPONENTEN, ETC.

DOMAIN DRIVEN DESIGN

- ▶ Etablieren einer allgegenwärtigen fachlichen und technischen Sprache
- ▶ Fördern von interdisziplinärer Kommunikation
- ▶ Schwerpunkt ist das modellieren fachlicher Zusammenhänge
- ▶ Kapseln von fachlicher Logik und technischen Abhängigkeiten

DOMÄNENMODELL (VEREINFACHT)

- ▶ Wertobjekte, Entitäten und Aggregate
- ▶ Assoziationen und fachlich/technische Ereignisse
- ▶ Serviceobjekte, Repositories, etc.

„FUNCTIONAL“ DDD & REACTJS

VALUE OBJECT
ENTITY

AGGREGATE

EVENTS

SERVICES



COMPONENT
(FUNCTION/CLASS)

COMPONENT
COMPOSITION

FUNCTION
(HANDLER)

CONTEXT
(PROVIDER)

REACTJS KONZEPTE UND ARCHITEKTUR

- ▶ Zustand vs. Zustandslos mit Smart- und Dumb-Components
- ▶ Komponentenorientierung und Komposition mittels Higher-Order-Components & Render-Props-Pattern
- ▶ Kontext und das Provider-Consumer-Pattern
- ▶ Asynchrone Kommunikation mit RESTful APIs, GraphQL, etc.
- ▶ Event-getrieben mit globalem Zustand durch Flux/Redux

STATELESS VS. STATEFULL COMPONENT

```
<Component text=„ABC“ />
```

RENDER

```
<Component text=„ABC“ />
```

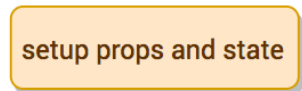
RENDER
USESTATE
RERENDER

```
const [isLoading, setIsLoading] = useState(true);
```

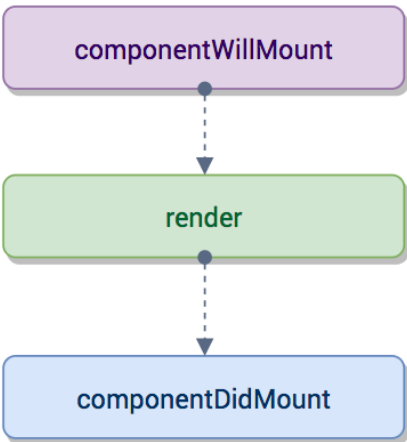
USEEFFECT

COMPONENT LIFECYCLE

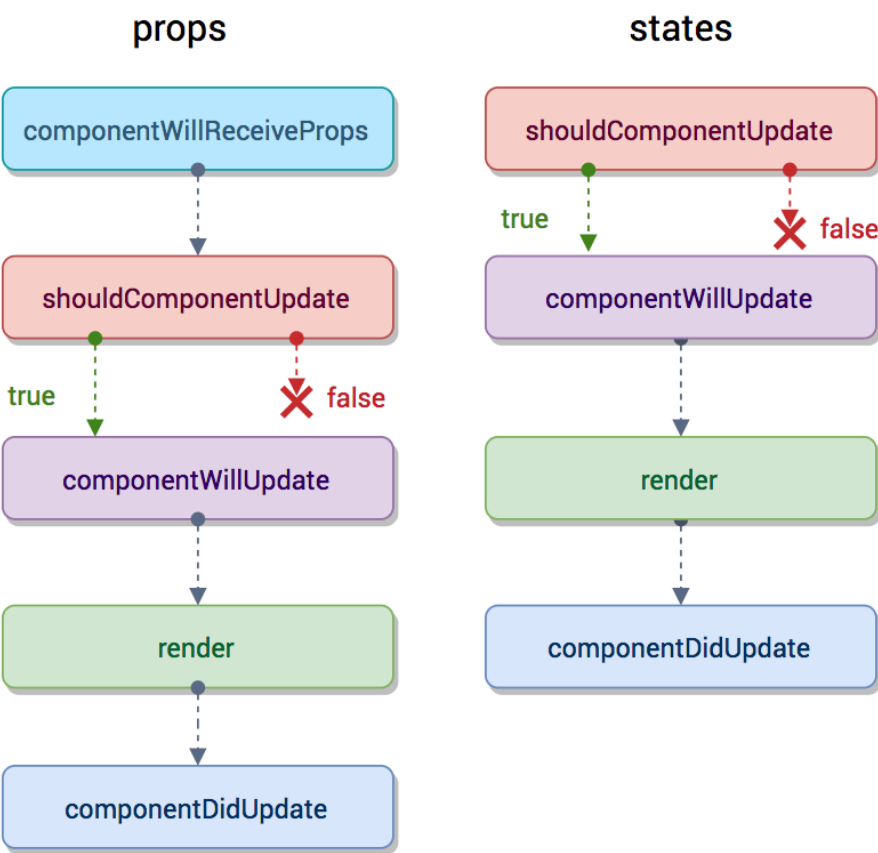
Initialization



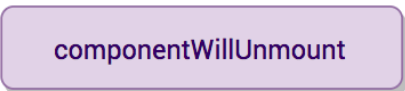
Mounting



Updation



Unmounting



COMPONENT LIFECYCLE MIT HOOKS

Tip

If you're familiar with React class lifecycle methods, you can think of `useEffect` Hook as `componentDidMount`, `componentDidUpdate`, and `componentWillUnmount` combined.

```
useEffect(() => {  
  | doSomething();  
}, []);
```

RENDER-PROPS PATTERN

```
<Component
```

```
  renderSomething={() => <ComponentA />
```

```
 />
```

```
<Parent>
```

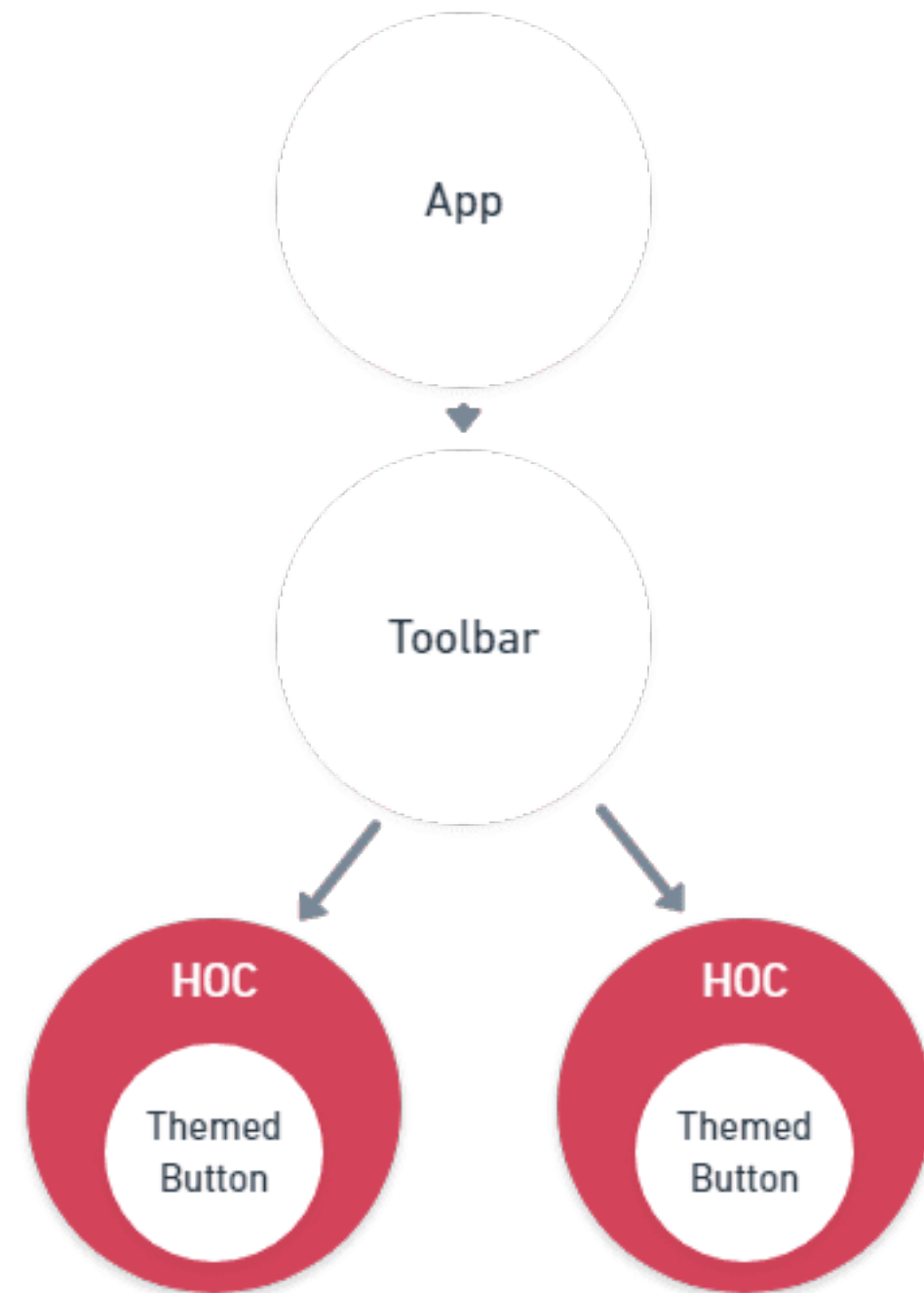
```
  {() => return <ComponentA />}
```

```
</Parent>
```

HIGHER-ORDER-COMPONENTS

COMPONENT

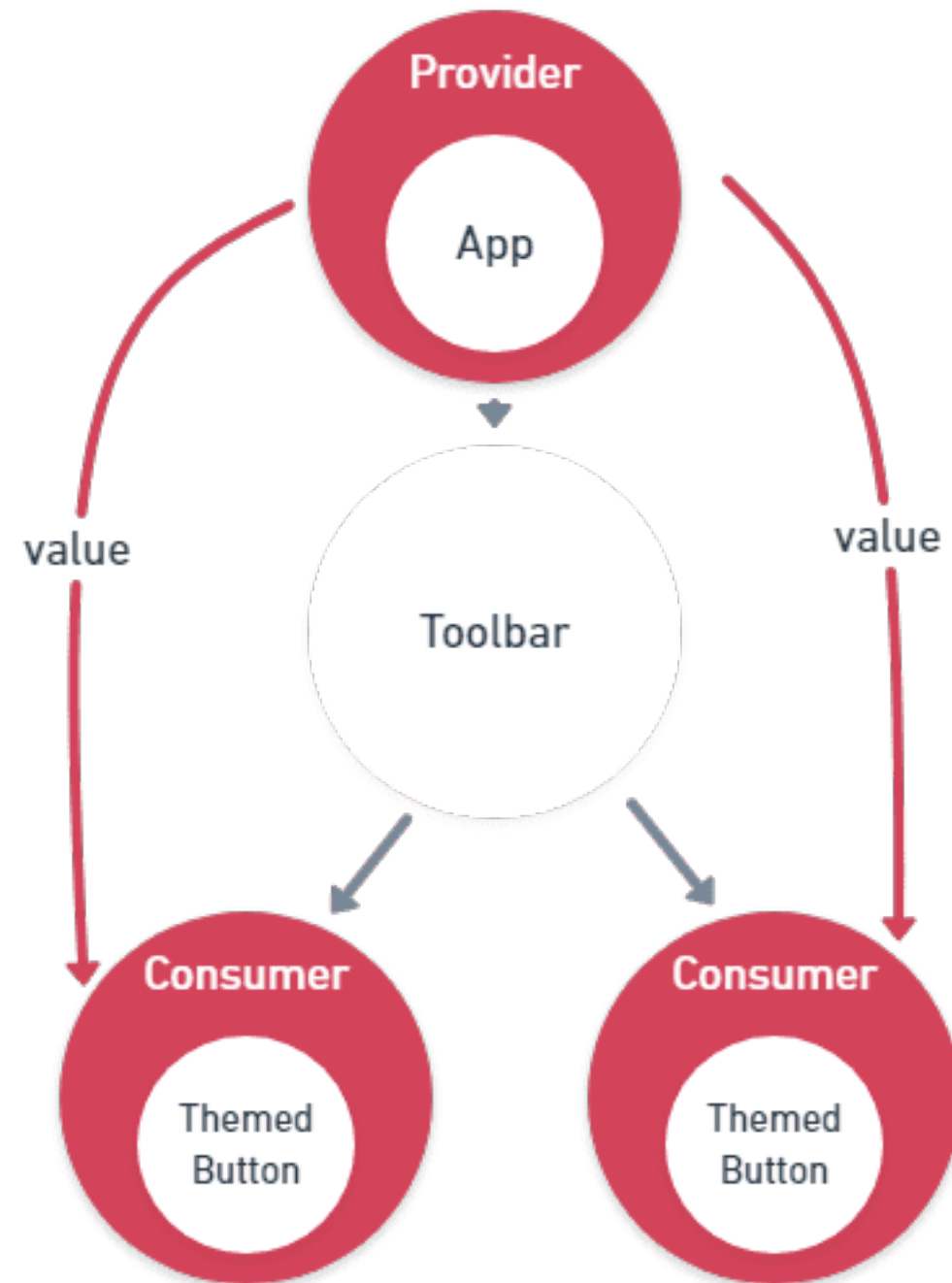
OWN STATE



CONSUMER-PROVIDER PATTERN

KONTEXT

SHARED STATE

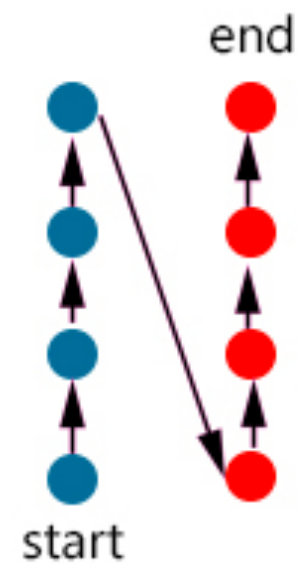


HOCS VS. PROVIDER

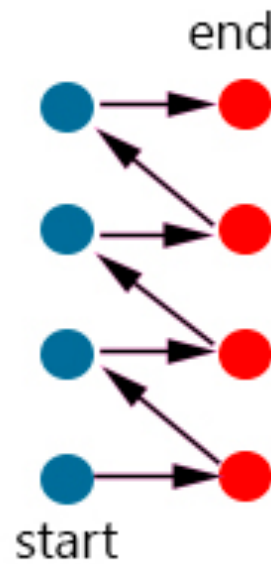
- ▶ Sind Daten/Zustand global? Provider
- ▶ Ist der Zugriff dynamisch? Provider
- ▶ Handelt es sich um statischen Zugriff? HOC

ASYNCHRONE KOMMUNIKATION

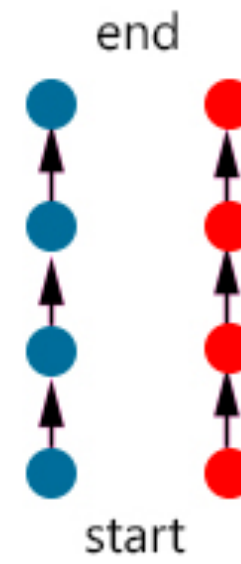
Sequential



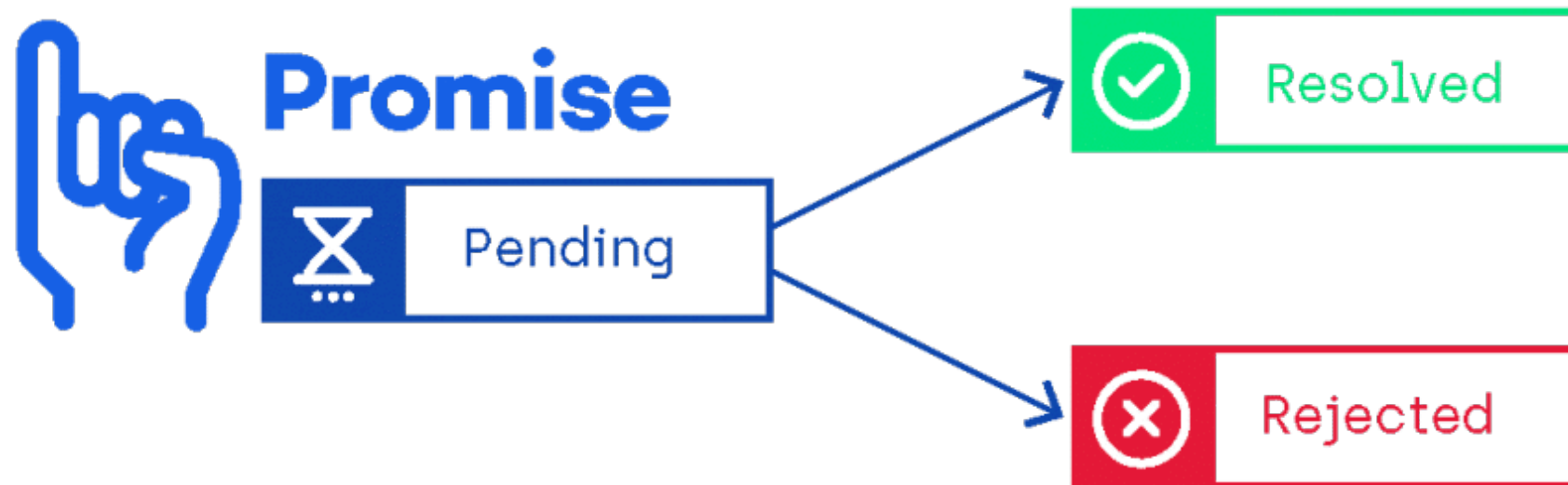
Concurrent



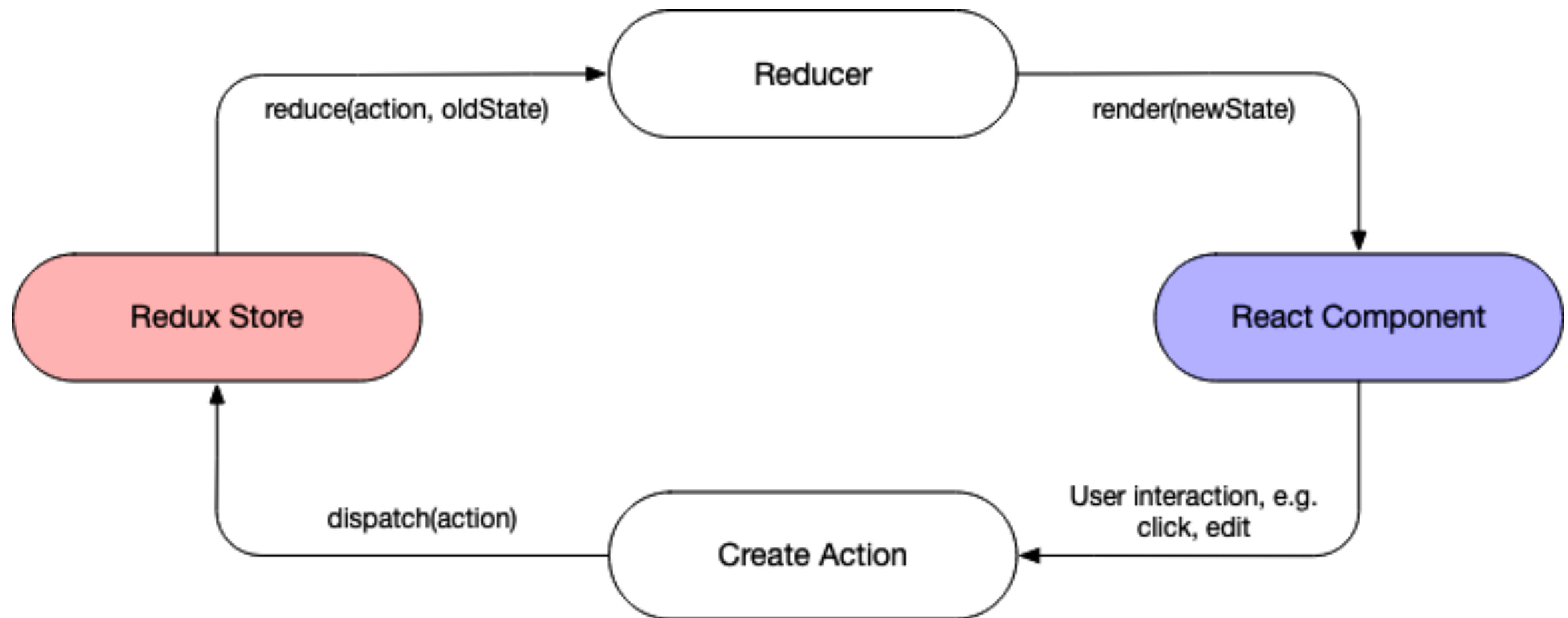
Parallel



ASYNCHRONE PROGRAMMIERUNG

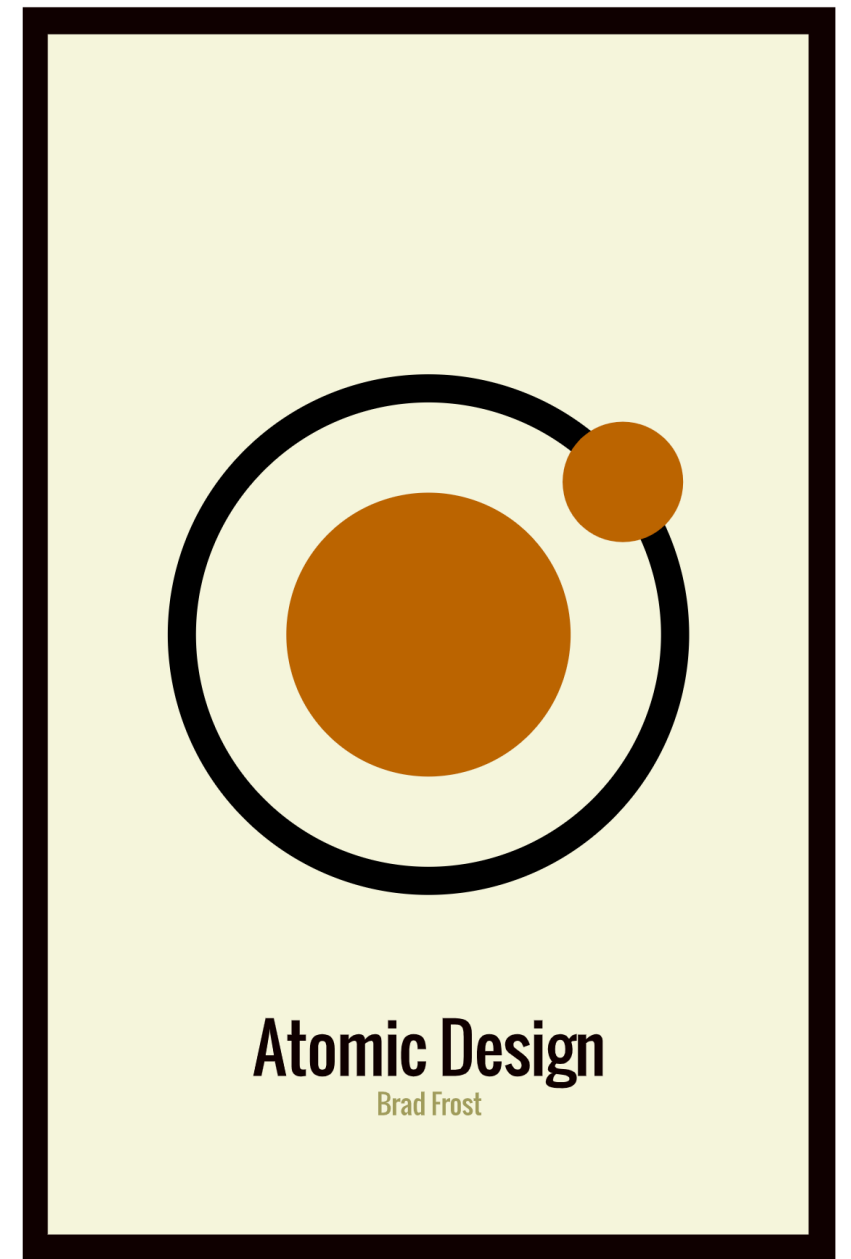


EVENT-DRIVEN MIT FLUX/REDUX



FRAGEN?

ATOMIC DESIGN



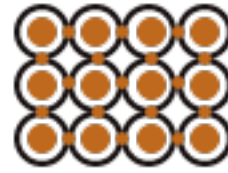
KOMPONENTEN



ATOMS



MOLECULES



ORGANISMS




TEMPLATES








PAGES

KOMPOSITION



ATOMS



>




Breadcrumb test


375 x 250

Property Name

Finance Company Name


Tracker Label







User Full Name

Category (Incomplete)

Category (Complete)






MOLECULES


Home

Breadcrumb > Breadcrumb


375 x 250


Property Name
Finance Company Name
Tracker Label
Tracker Label







User Full Name

Category (Incomplete)

Category (Complete)







ORGANISMS

Home


Breadcrumb > Breadcrumb


Property Name
Finance Company Name
Tracker Label
Tracker Label







User Full Name

Category (Incomplete)

Category (Complete)







TEMPLATES

Home


Breadcrumb > Breadcrumb


Property Name
Finance Company Name
Tracker Label
Tracker Label




User Full Name


Category (Incomplete)

Category (Complete)





User Full Name

Category (Incomplete)

Category (Complete)







PAGES

Home


Properties > Bryant St.


Bryant St.
Alterra Home Loans
Borrowers
Lender




John Marder (Me)

Identity

Assets

Jane Peachcore

Identity

Assets

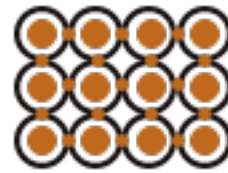
KOMPONENTEN



ATOMS



MOLECULES



ORGANISMS



TEMPLATES



PAGES

ELEMENT

KOMPOSITION

KONTEXT
STATE
SEITENEFFEKTE

LAYOUT & SKELETON

INTEGRATION

STYLE GUIDE GUIDE

STYLEGUIDEGUIDE

Getting Started

Guidelines



Styles



Components



Utilities



Page Templates

Downloads

Support

Contribute

Style Guide Guide

A boilerplate for creating superb style guides

The homepage of a style guide should provide high-level information around what the design system is, what benefits it provides, who it's for, and how to get started with it. Like any good index page, it should provide clear navigation to key parts of the website.

Get Started

Design System Benefits

Explain how the design system benefits users and the business. For inspiration, check out over 180 examples of design systems at [Styleguides.io](https://styleguides.io).

Contributing info

If it's desirable to have people from across the organization contribute to the design system, linking to the [contributing.page](#) from the homepage could be a good idea.

[About](#)

[Roadmap](#)

[Release History](#)

[Contribute](#)

TOOLS

- ▶ UI Environment/Playground - **Storybook** (<https://storybook.js.org/>)
- ▶ Design Systeme wie **Material-UI, Fabric-UI, Bootstrap, etc.**
- ▶ Erstellen von Style Guides - z.B. <https://bradfrost.github.io/gatsby-style-guide-guide>

ZUSAMMENFASSUNG

- ▶ Schwerpunkt und Ziel liegt auf UI/UX Komponentenorientierung
- ▶ Isolation, Struktur und Organisation von Komponenten
- ▶ Modularer Aufbau von Komponenten
- ▶ An den Grundbegriffen der Naturwissenschaften orientiert
- ▶ Entwurf und Umsetzung von Komponenten-APIs

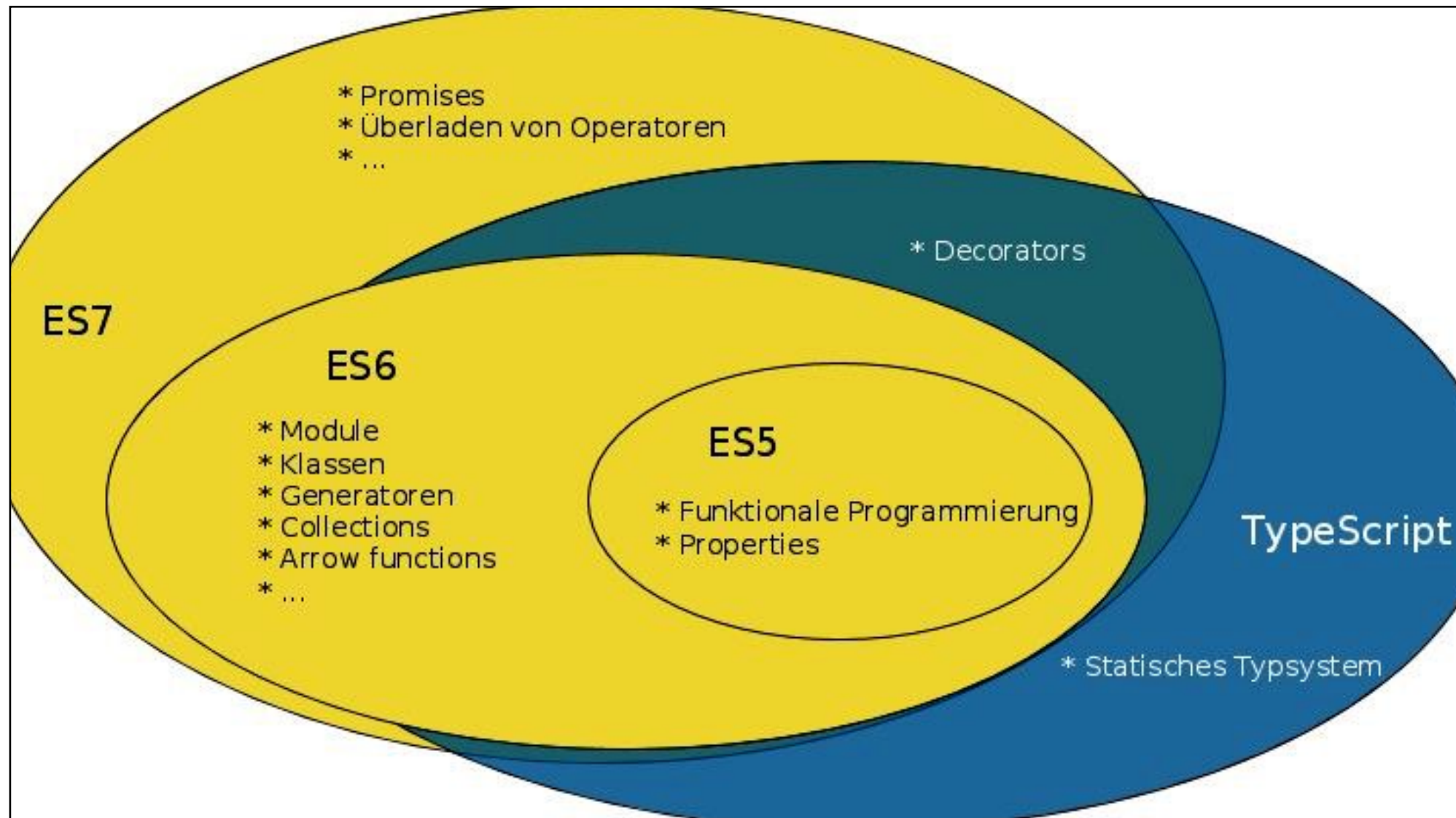
BEISPIELE

- ▶ Material-UI (<https://material-ui.com>)
- ▶ Fabric-UI (<https://developer.microsoft.com/en-us/fabric>)

FRAGEN?

JAVASCRIPT
ECMASCRIPT
TYPESCRIPT

UI/UX ENTWICKLUNG MIT REACTJS

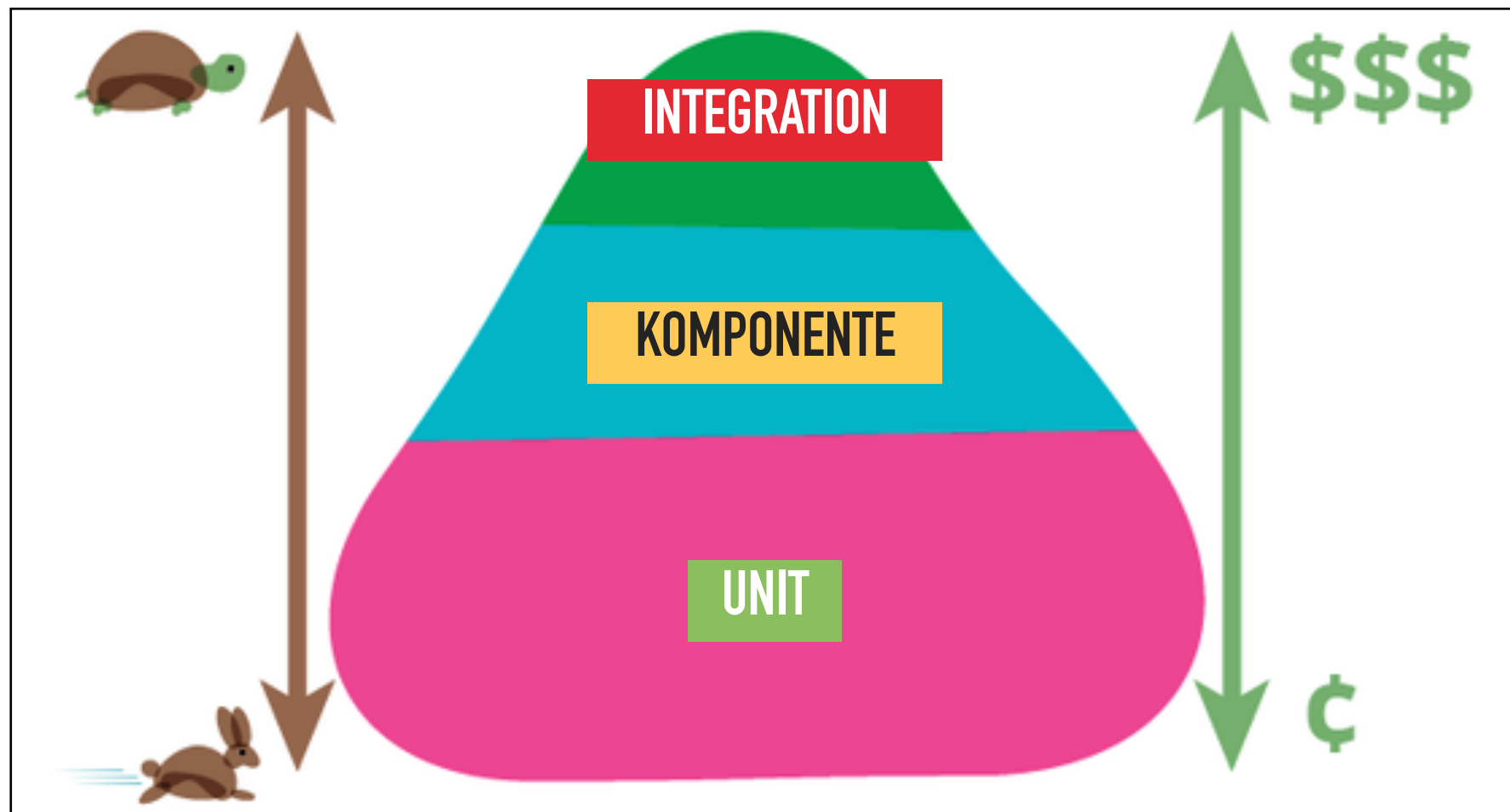


FRAGEN?

TESTING

GESCHWINDIGKEIT
AUFWAND

KOSTEN
WERT



REACTJS TEST-TOOLS

UNIT
TESTS

JEST

COMPONENT
TESTS

JEST

UI/UX
TESTS

PUPPETEER