Thinking in React – Components

# How to split Components in UI?

* Based on SIZE
  + If a COMPONENT does to many things, we NEED TO BREAK IT INTO MULTIPLE COMPONENTS
  + If a component needs 10-20 props to configure the component, it’s way to big
  + Hard to Reuse
  + Complex code, hard to understand

## The goal is to strike the RIGHT balance between too SMALL and too BIG

# 4 Criteria for Splitting UI into Components

1. Logical Separation of content/ layout
2. Reusability
3. Responsibilities/ complexity
4. Personal Coding Style

# Framework for building Components

1. Start with a relatively big component, then split it into smaller components as it becomes necessary.

## Logical Separation

* If the component contains pieces of content or layout that **DON’t BELONG TOGETHER** 
  + Create a NEW COMPONENT

## Reusability

* Is it possible to reuse part of the component ? 🡺 CREATE A NEW COMPONENT
* Do you want or need to reuse it? 🡺 CREATE A NEW COMPONENT

## Responsibilities/ Complexity

* Is the component doing too many different things? 🡺 Create a new component
* Does the component rely on too many props? 🡺 Create a new component
* Does the component have too many pieces of state and/or effects? 🡺 Create a new component

## Personal Coding Style

* Do you prefer small functions / component ? 🡺 Create a new component

# Component Categories

## Stateless/ Presentational Components

* These don’t have any state
* They receive some props and just present the data or other content
* Quite small
  + Logo
  + Num results
  + Movie
* Usually reusable

## Stateful Components

* HAVE STATE
* Can still be reusable

## Structural Components

* ‘Pages’ ‘Layouts’ or ‘Screens’ of the app
* Result of **COMPOSITION**
* can be HUGE and non-reusable

# Prop Drilling

* the process of passing some prop through multiple Nested Components until it reaches the desired Component

# Component Composition

* WE can Pass Another WHOLE COMPONENT to another component using the {children} prop
* A screenshot of a computer program

  Description automatically generated
* The main difference is that :
  + Using Component
    - The Modal cannot be reused since the ‘Success’ component is part of it
  + Component Composition
    - The Modal component can be reused since it will be able to receive any other component as a children, so it’s not tied to the ‘Success’ component

### We use component Composition to create HIGHLY REUSABLE COMPONENTS and to AVOID PROP DRILLING (great for layouts)

### A screen shot of a computer program Description automatically generated

# Props as an API

* In our situation, with small apps, we are the Consumers and the Creators
* In a big application, the Creator of the Component might not also be the Consumer of the COmponent
* There are 2 types of Component USERS

### Component Consumer

* This is the user that will use the already created component using an APP
* He is using the Component and it’s props as an API

### Component Creator

* The Creator is the designer of the component
* He needs to think and design the props and the logic so that every consumer can use the component
* He needs to make the component highly reusable
* For the PROPS API
  + You need to find the right balance between to many Props or to little Props

# PropTYpes – not used anymore since we use TS

* With PropTypes we can define what type of prop are we expecting the **CONSUMER**  to pass into our Prop API
* We define the PropTypes
* A screen shot of a computer

  Description automatically generated
* -

\

* Based on these PropTypes, if the user passes something different, the REACT will throw an error in the console



A screenshot of a computer program

Description automatically generated