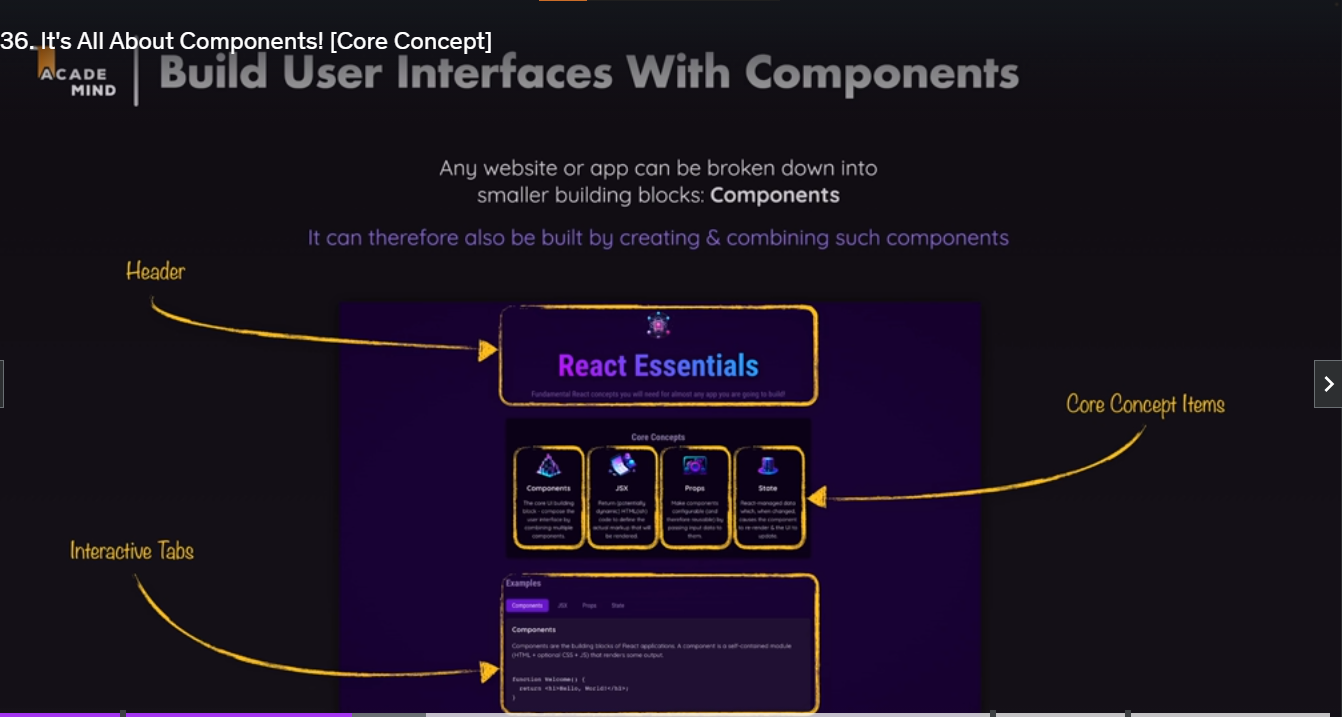
# React Essentials-Components, JSX, Props and State

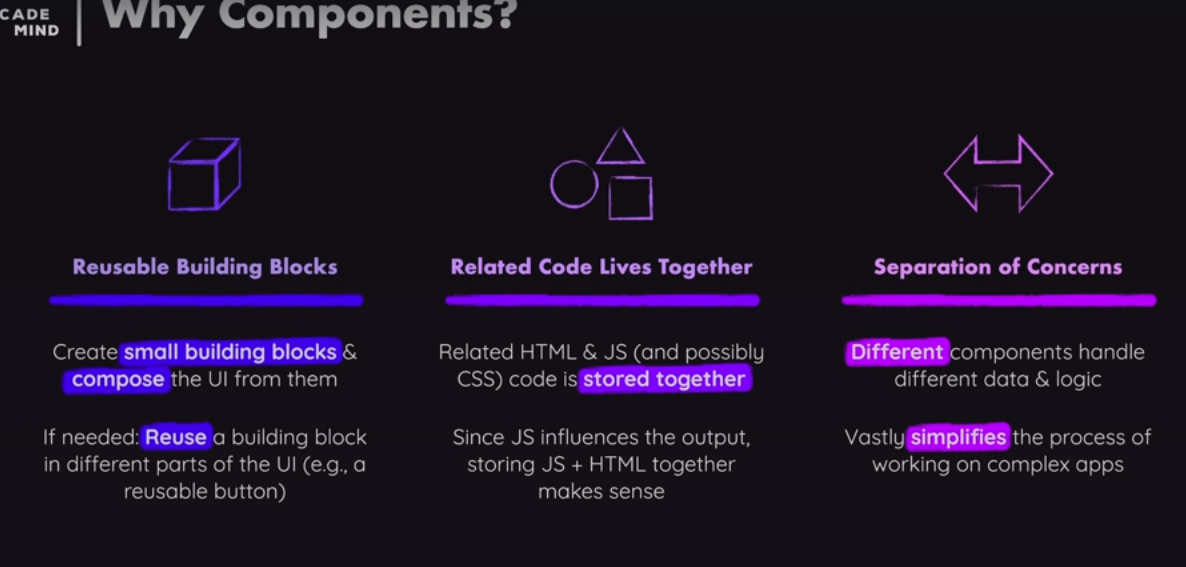
## Components:

Components are UI building blocks. They are the reusable building blocks which a react developer can create and which is then combined to build the overall user interface.

***The react apps are build by combining components.***

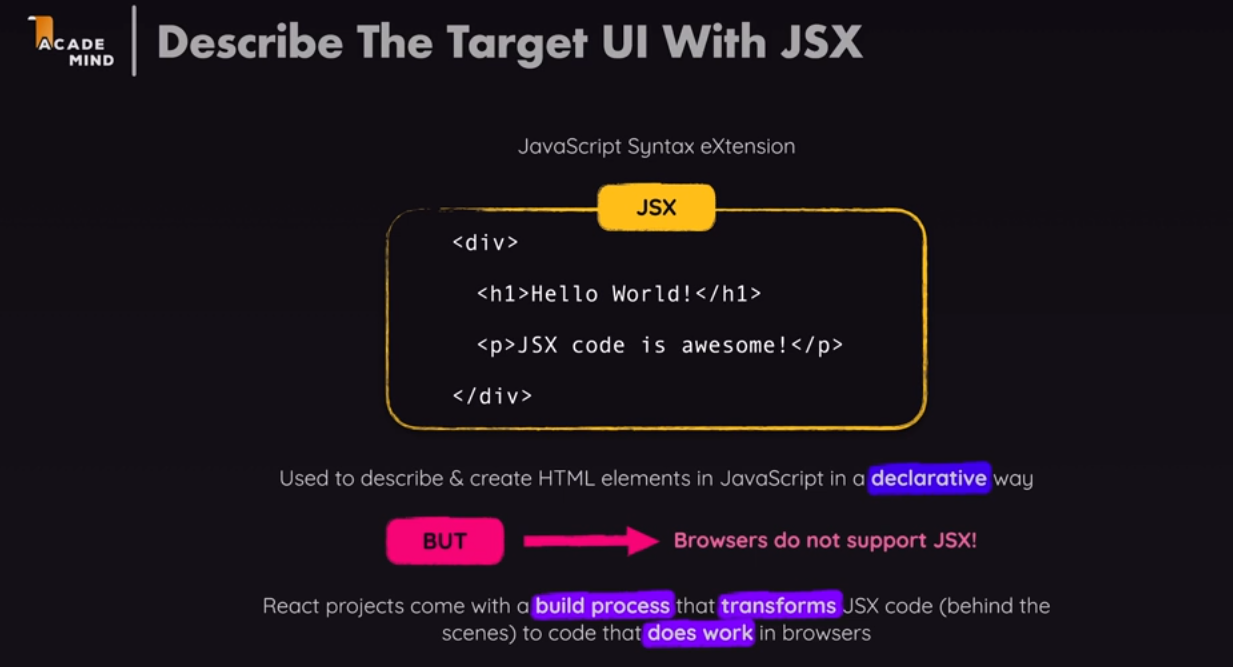


The components are created by wrapping the HTML, CSS and Javascript codes defines the part of the UI. This allows the developer to split up complex user interfaces into smaller and manageable parts, which can be reused in different places.

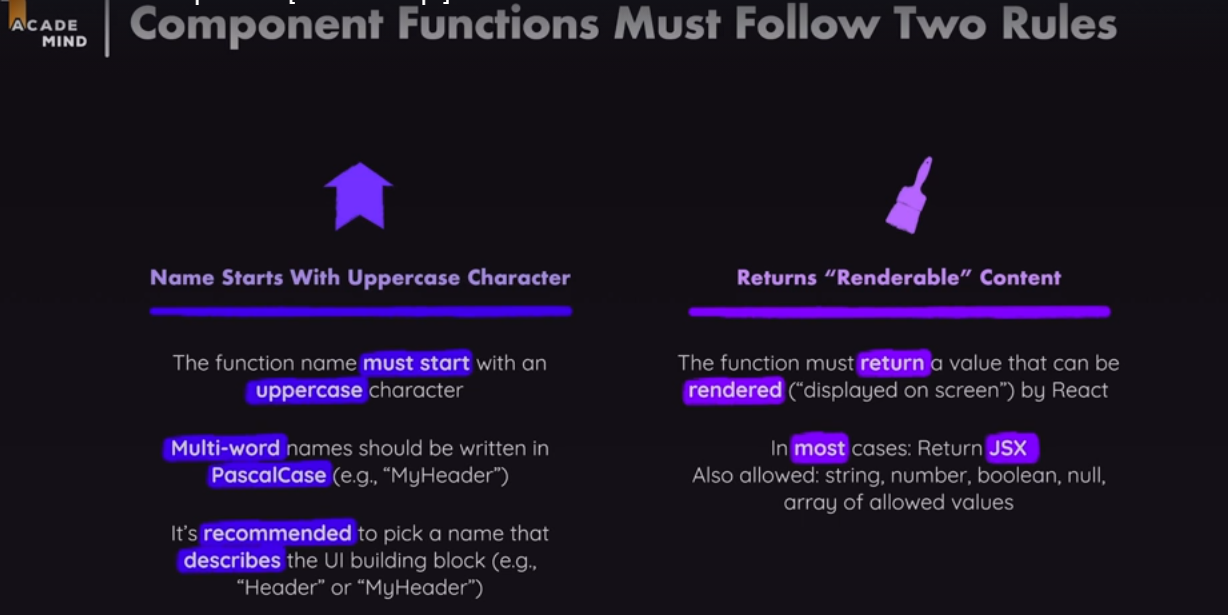


## JavaScript Syntax Extensions

This extension allows developers to describe and create HTML elements, by writing HTML markup code inside of java script files. Since, react is about describing and creating user interfaces, this is a very useful feature.



This .jsx is not supported by the browsers. Instead, the code written is transformed behind the scenes by the development servers before it reaches the browser.

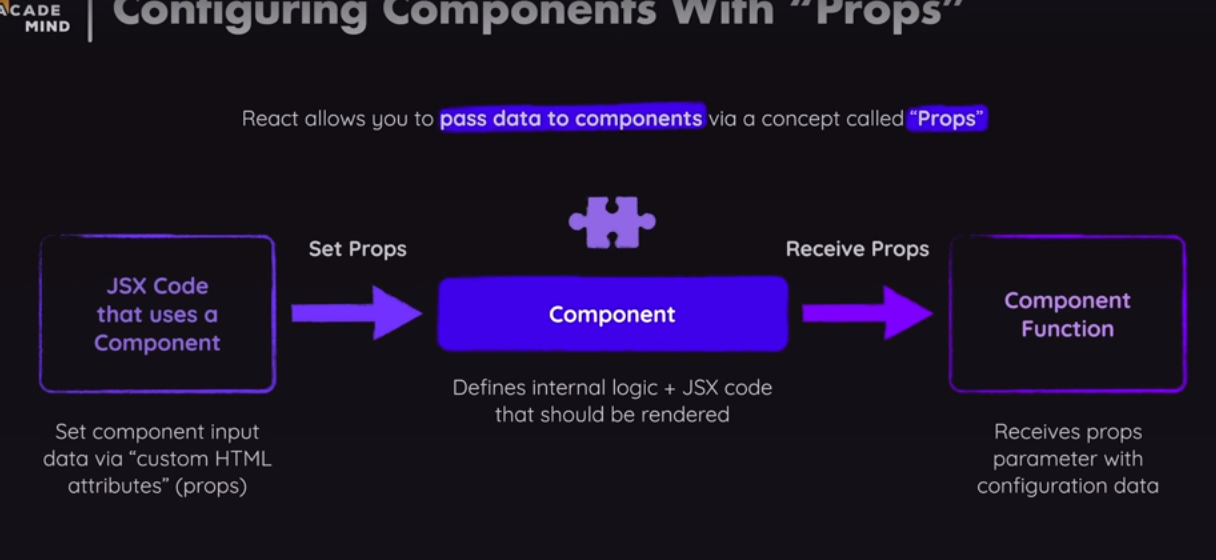


## Making Components Reusable with props

The main advantages of components are they are reusable.

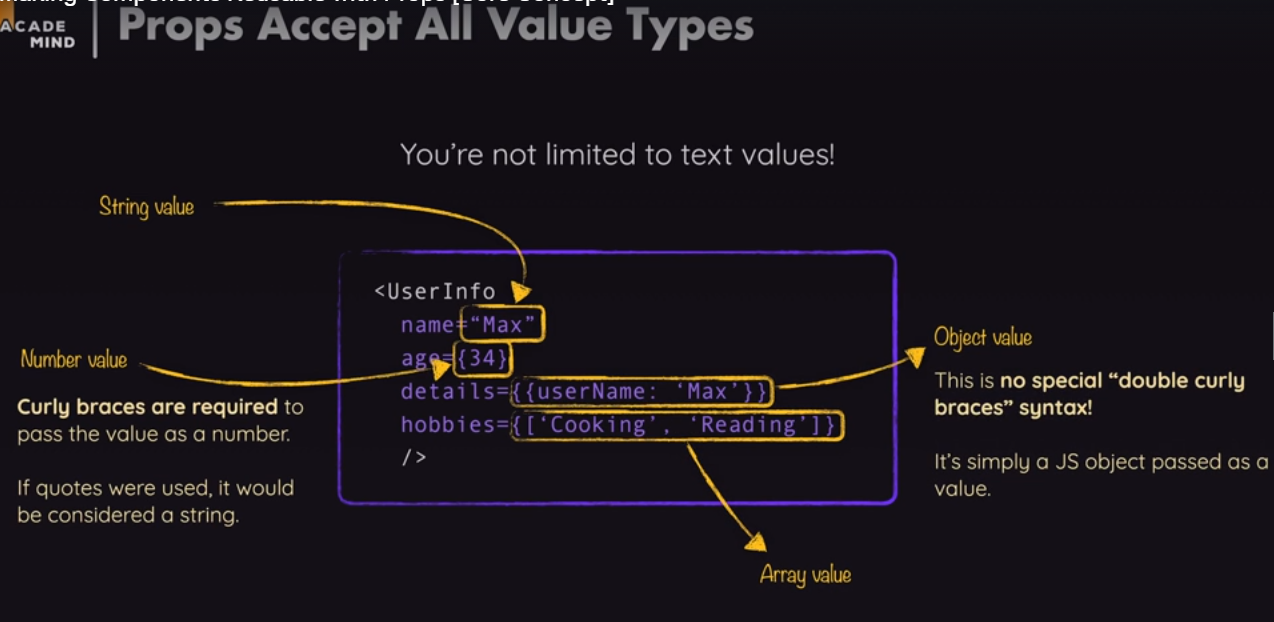


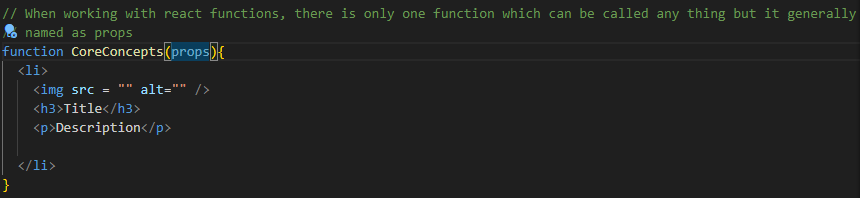
There is another crucial concept that is related to components which is known as **props.**

****

The concept of prop is being able to pass data into components and then to use them.



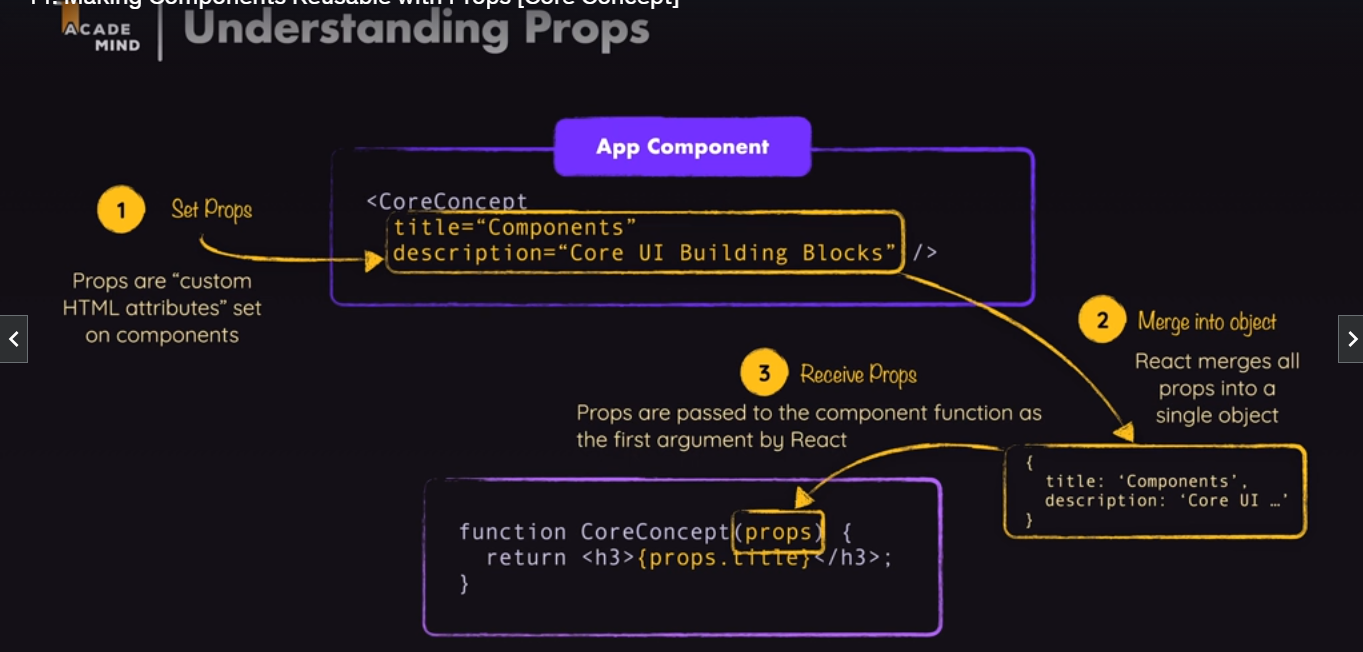


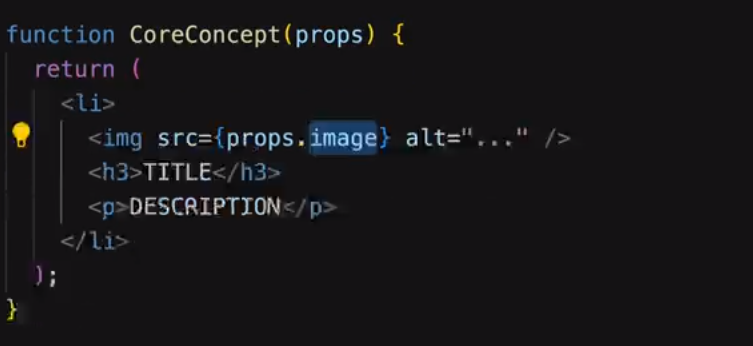


The overall react concept is known as the props, therefore, the name is given as props in the **CoreConcept** parameters. The **props** parameter will be set by react, because the react will execute the function **CoreConcepts.** The developer is not calling the component functions by themselves in the code. Rather, they are used as the HTML elements and under the hood react calls the actual functions. Therefore, react passes a value for the given props parameters to the function when it calls it and the value that will be passed for this function will be an object. It will be an object that has the value of the props in the form of key value pairs.

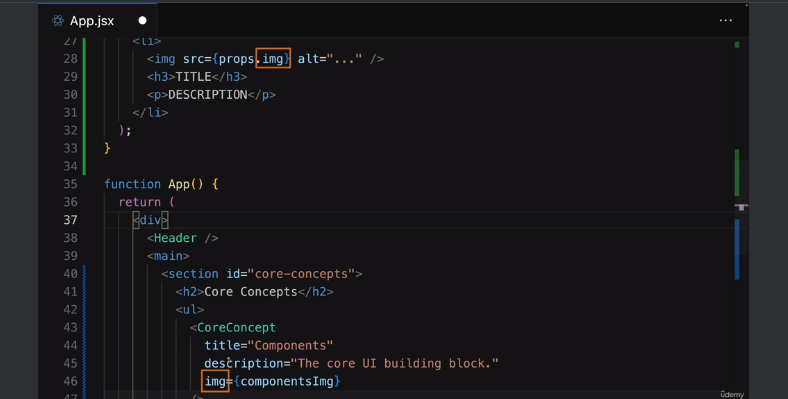


In the given figure, the *title is the key*  and its corresponding value is the “*Components”.* Therefore, the custom attributes acts as keys and the values for this keys are grouped as *values.*





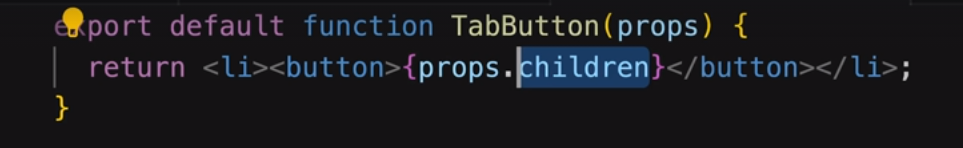
***props.img*** is used to get hold of the image ***ComponentsImg.***

******

## Component Composition: The special ‘Children Prop”[Core Concept]



The ***”Components”*** between the TabButton is not been displayed on the page as the react does not knows where to output it. Every custom component will receive props even if no attributes are set, react always returns the prop object which will be empty in nature. However, the props received is not actually entirely empty. There is one prop which is always present which is the built-in *‘children props’.*

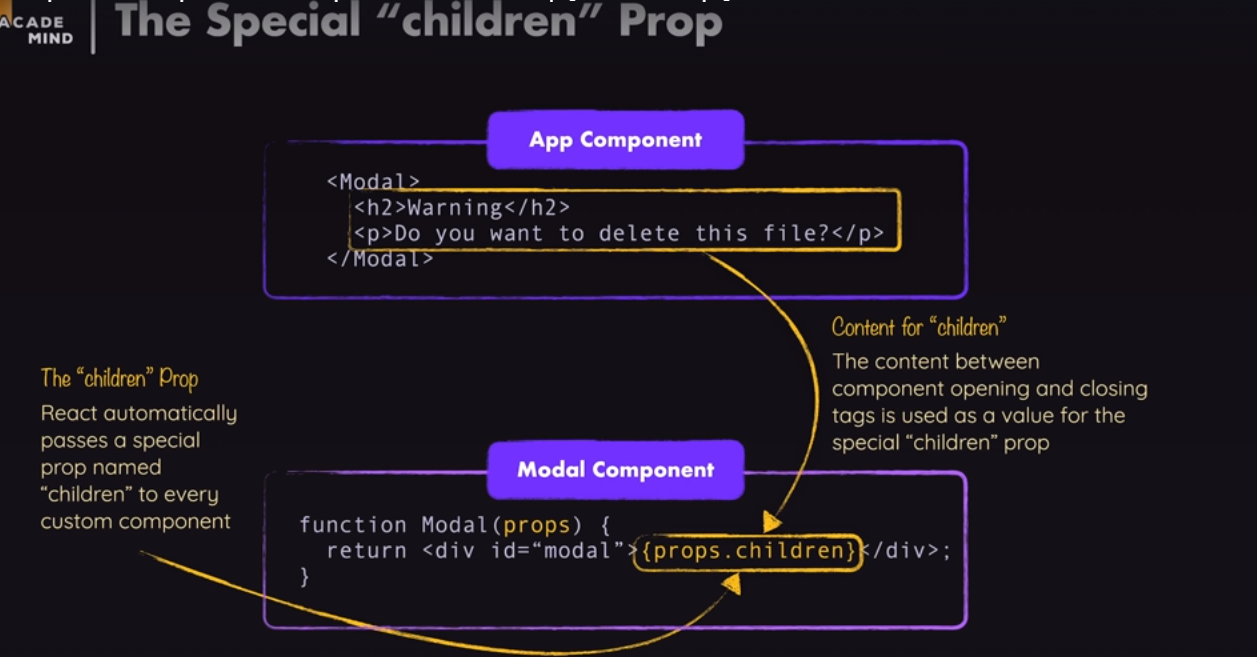
**

The children prop is set by the react and therefore it is not set with the help of attributes.

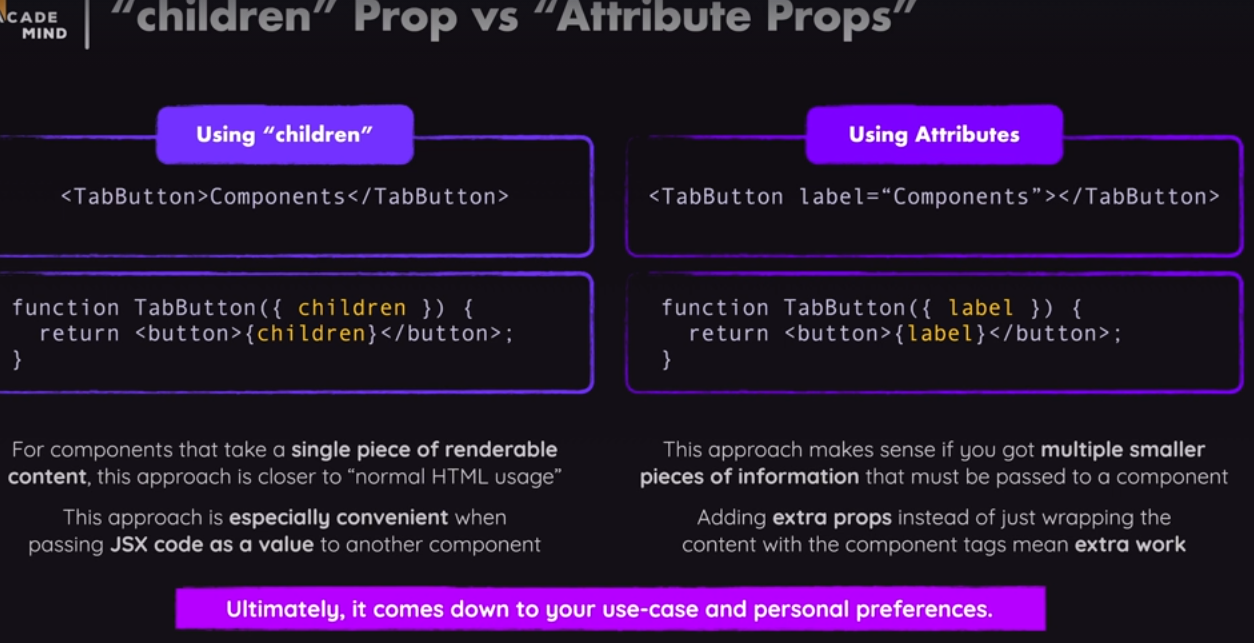




The children prop refers to the content between the component text .



So, the way of building components, where the components can wrap other components or other contents is known as component composition.



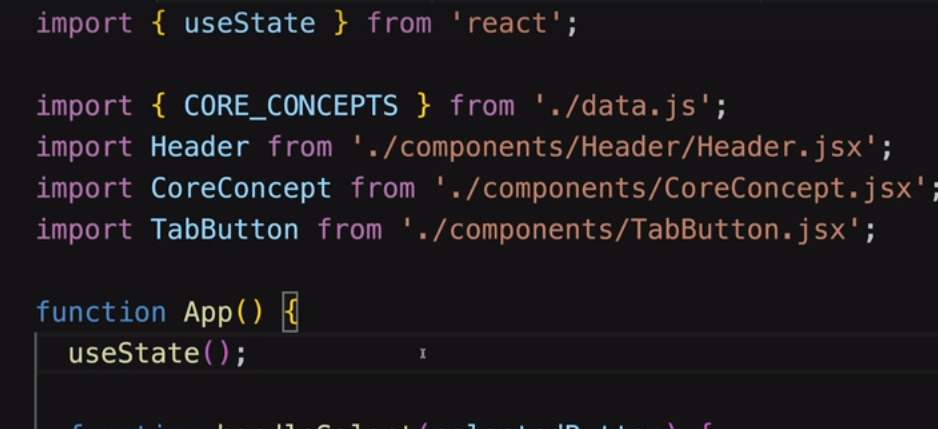
## Reacting to Events [Core concept]

To make a button active in react we use variety of special types of props. One of such props is known as onClick and this attribute takes a function as its value.



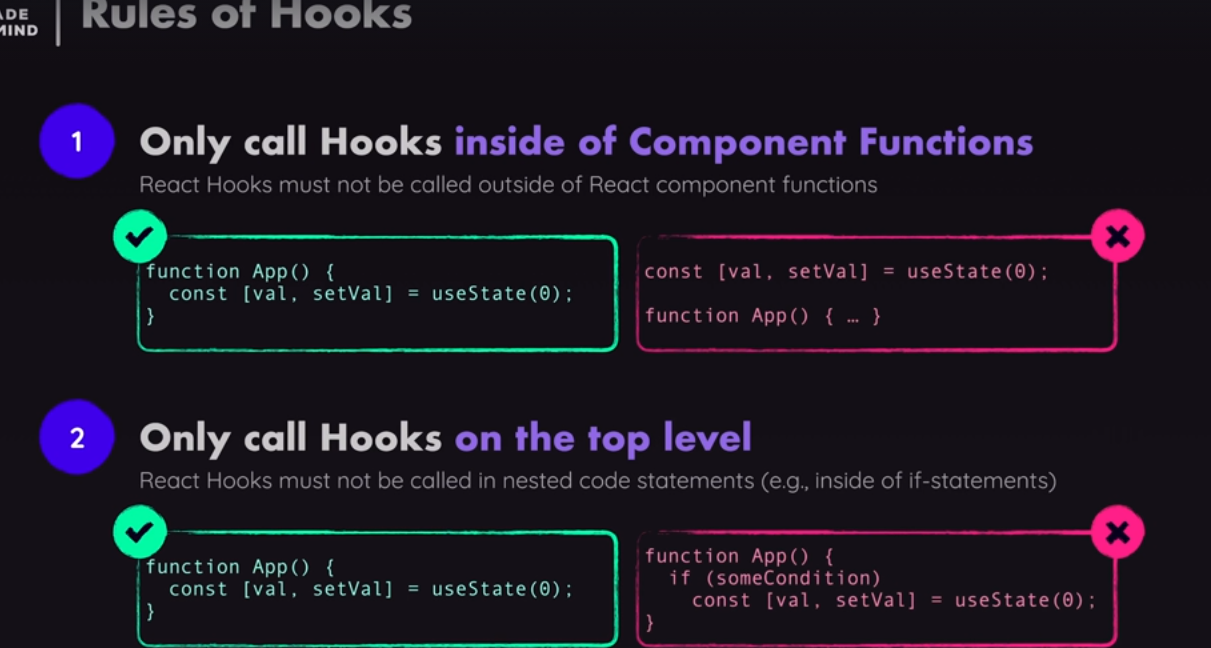
## Managing States and Using Hooks

State: A state is all about registering variables that are handled by react and are updated with the help of a function that is provided by react that will also tell react that data changed, and therefore causes the UI to get updated. These special variables are created with the help of a special function that must be imported from the react library. The function is known as the ***‘use state’.*** This is also known as the react hook. All the functions that starts with ***‘use’*** are react hooks. They are technically regular functions, but, they must only be called inside of react component functions or inside of other react hooks.

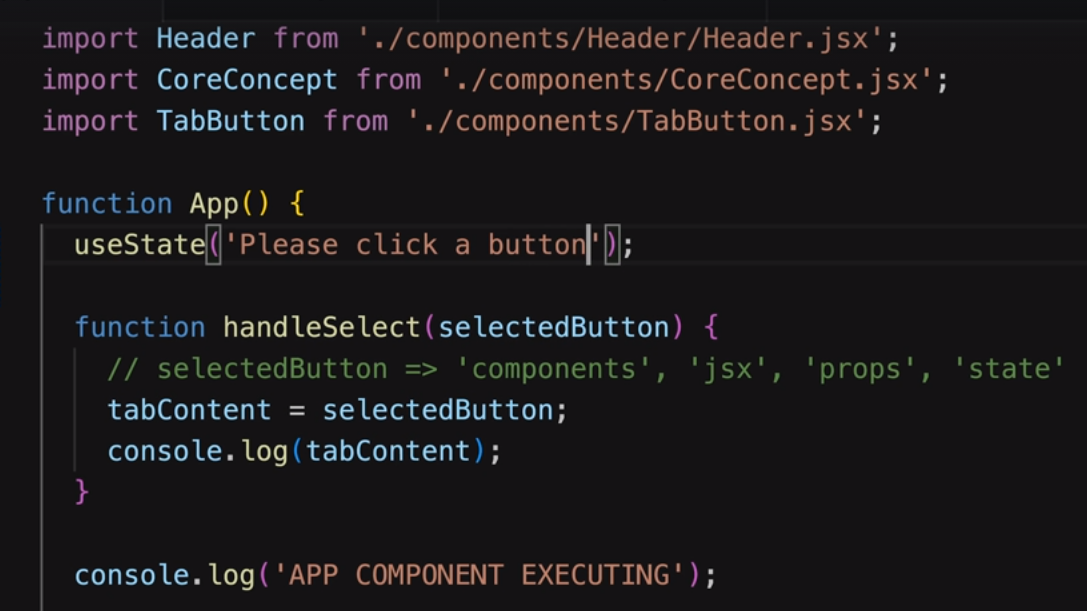


The useState() must be called at the top level of the component functions.

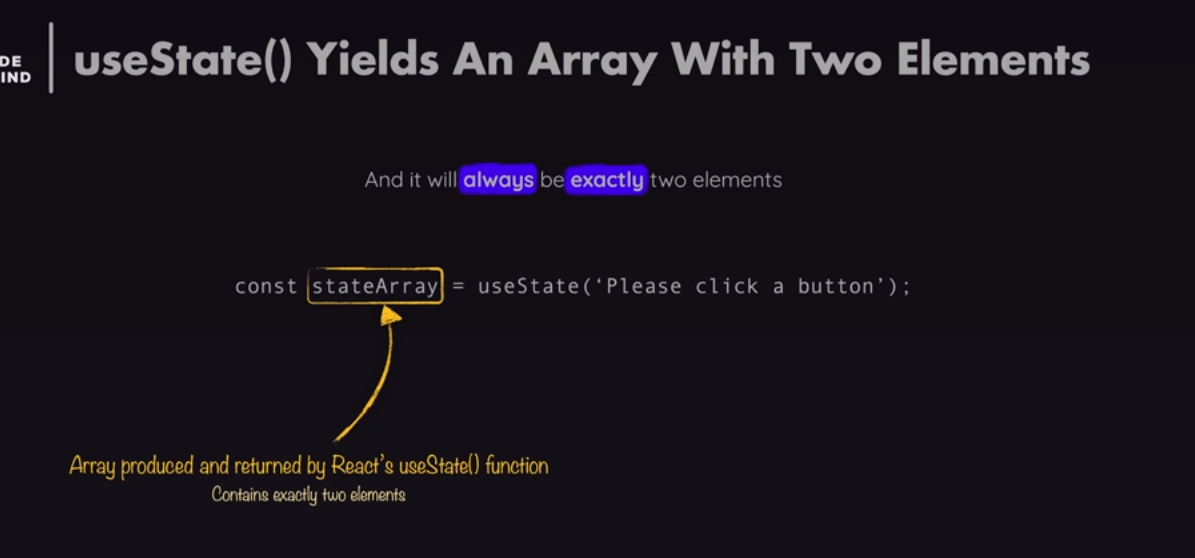
There are two important rules while using hooks.

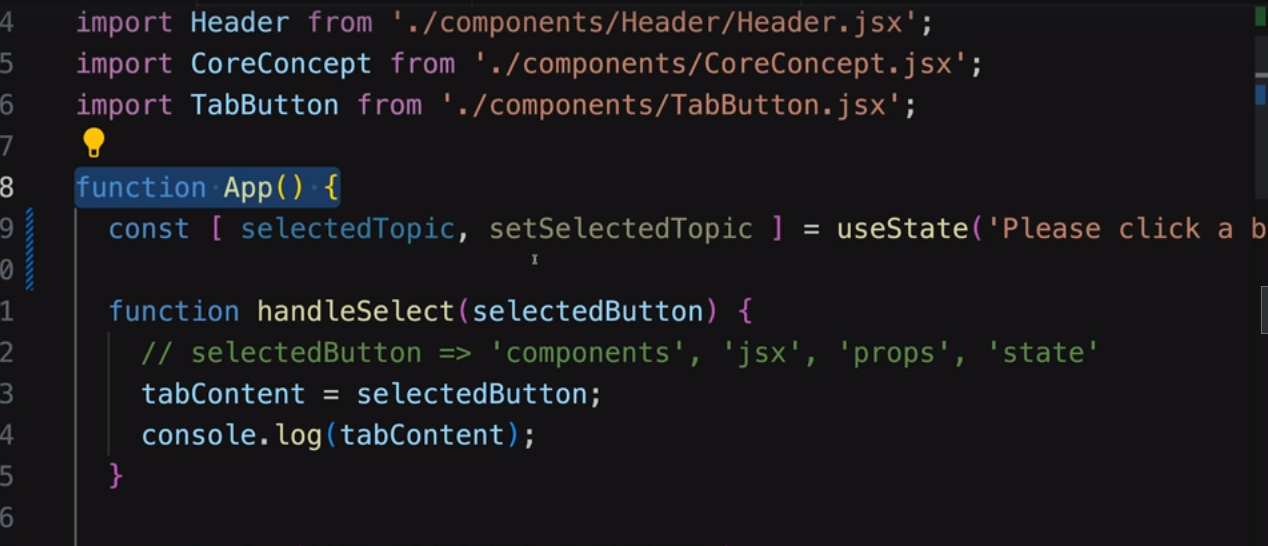


The use state allows to manage some components specific states, that could be some data which when changed triggers the component functions which the hooks belong to re-execute. The useState accepts an argument which is the default value you want to store, which could be the value when the react is first rendered.



The use state then returns a value which is then stored in a constant or a variable which is in the form of a array that includes exactly two elements.





The first element ***‘selectedTopic’*** consists of the current data snapshot for the App() component execution cycle. So, when the component function is executed first, it will be the initial value. When it is executed after the first execution, it contains the value of the executed value. Therefore, the first element is the first data we are managing. The second element, ***setSelectedTopic,*** will always be a function. The function is provided by the react that can be executed to update the state. But, in addition to updating the stored value, calling the special function ***setSelectedTopic,*** will also tell react that the component function App() must be executed again.

The ***‘selectedTopic*** is recreated every time the the App() component executes.

