**React** is all about “components” because all user interface in the end are made up of components. With React, you **define the target UI state(s)** – not the steps to get there! Instead, React will figure out & perform necessary steps. (Declarative programming)

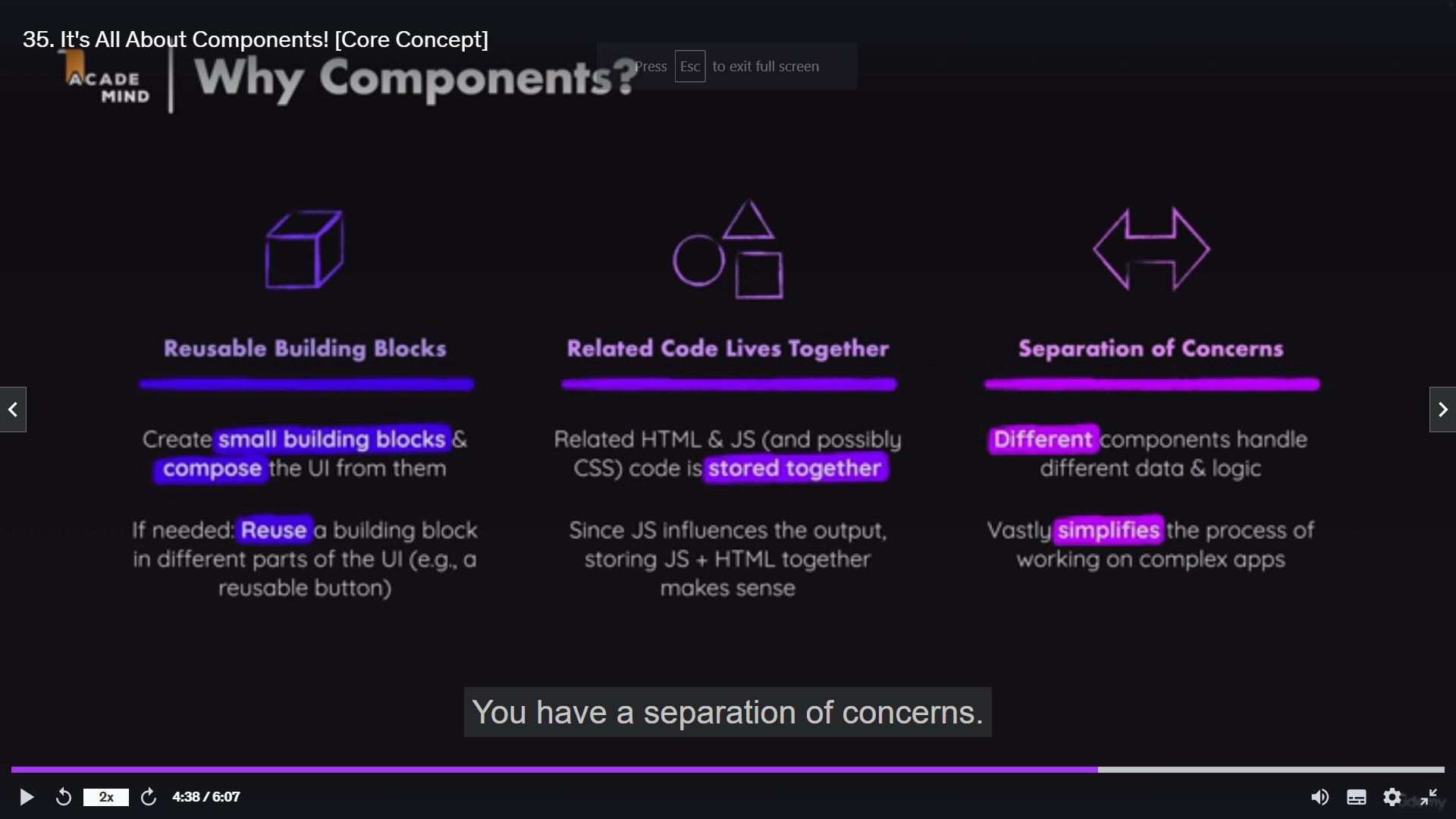
React project uses a build process:

* Raw, unprocessed React code **won’t execute** in the browser because React uses JSX feature.
* In addition, the code would **not be optimized for production** (e.g. not minified)

**Components** are reusable building blocks in your user interface (just a combination of HTML code, CSS styling, JS logic).

Ex:

Reason why we use components:



Graphical user interface, application

Description automatically generatedHow components are built:

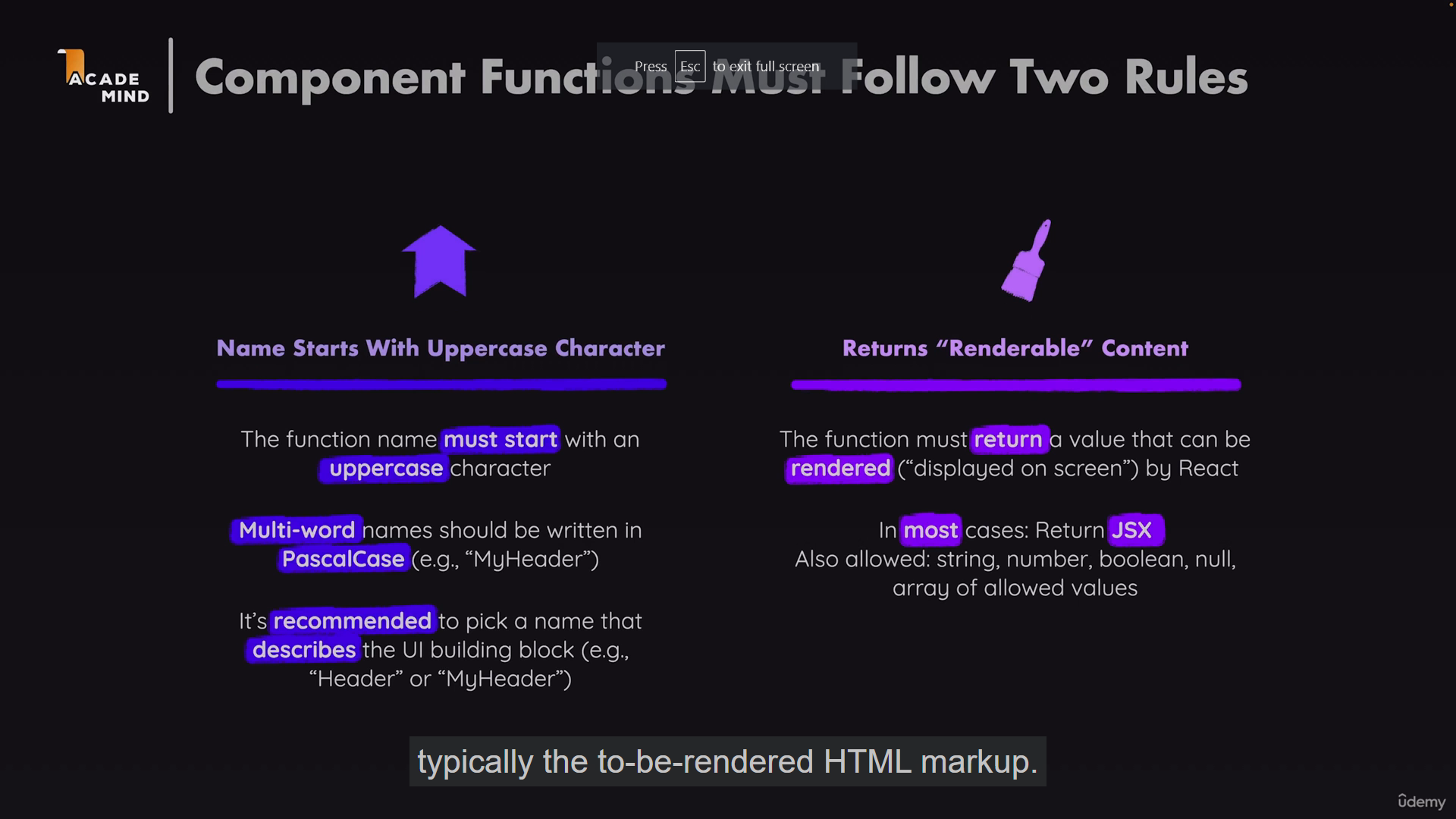
**React code** is written in a “declarative way”. With React, we will not tell that a certain HTML element should be created and inserted into a specific place. We will always define the desired end state, and React’s job to figure out which elements on actual webpage might be added or removed or updated.

Graphical user interface, website

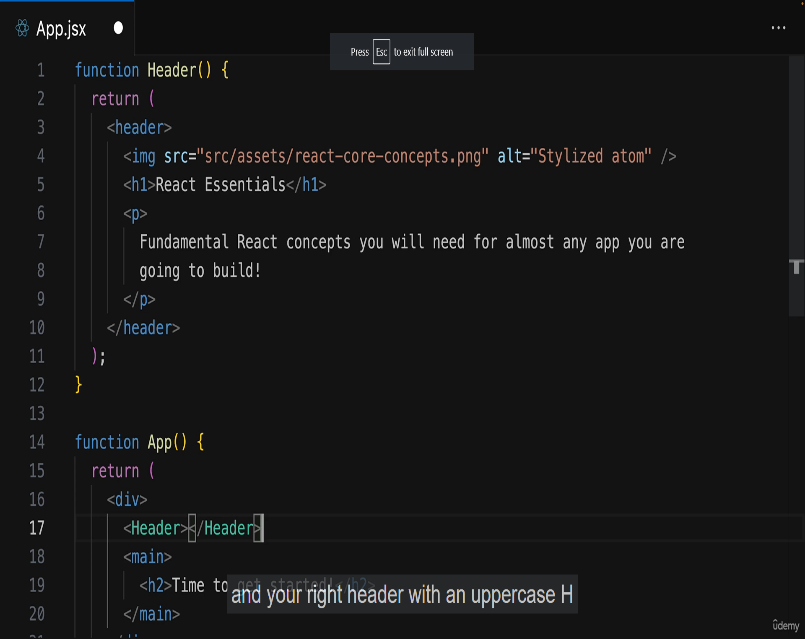
Description automatically generated

**JSX (JavaScript XML)** is basically HTML code inside of JavaScript.

Component function rules:

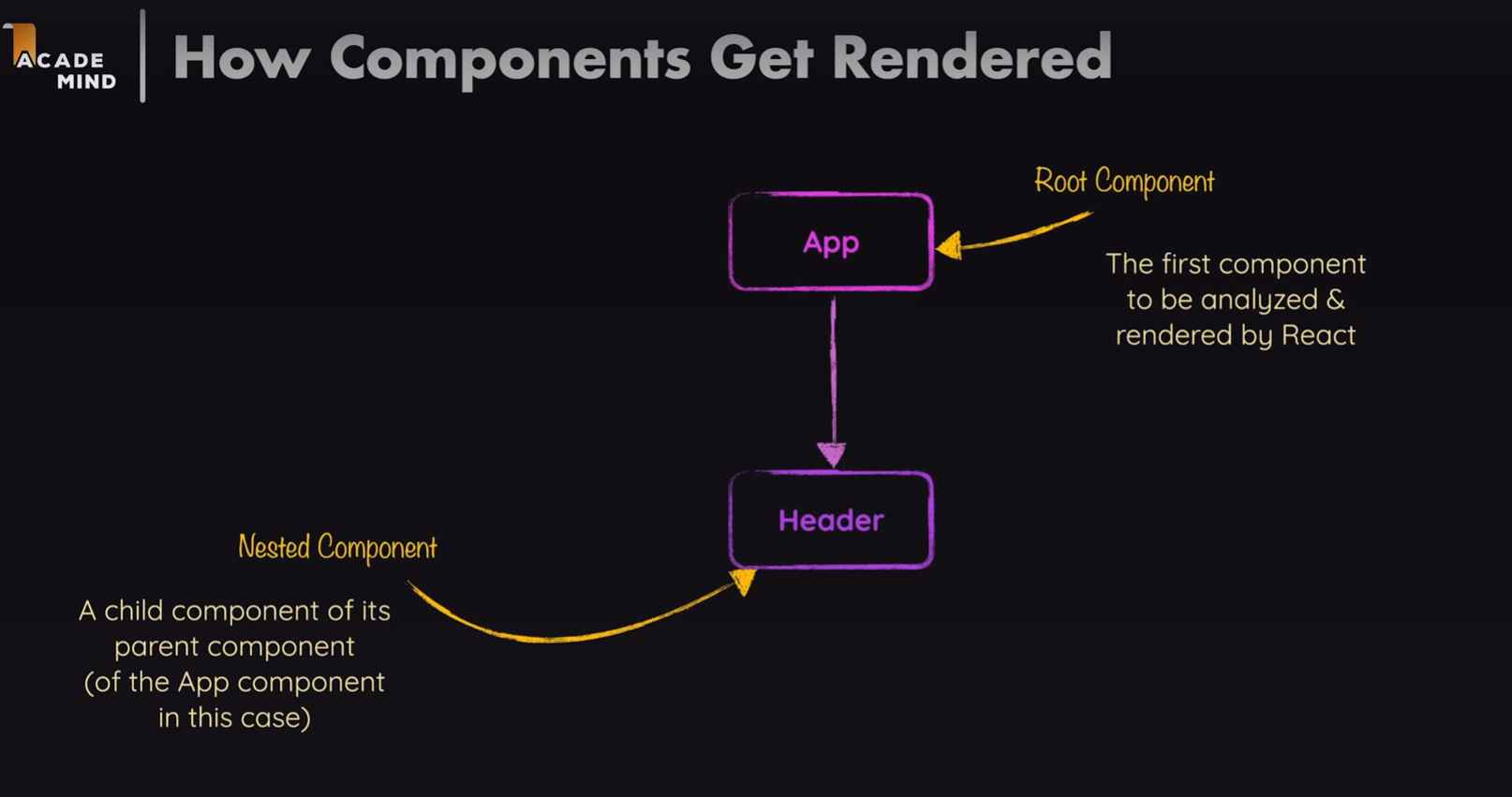


With React, we basically create component by building up the component tree

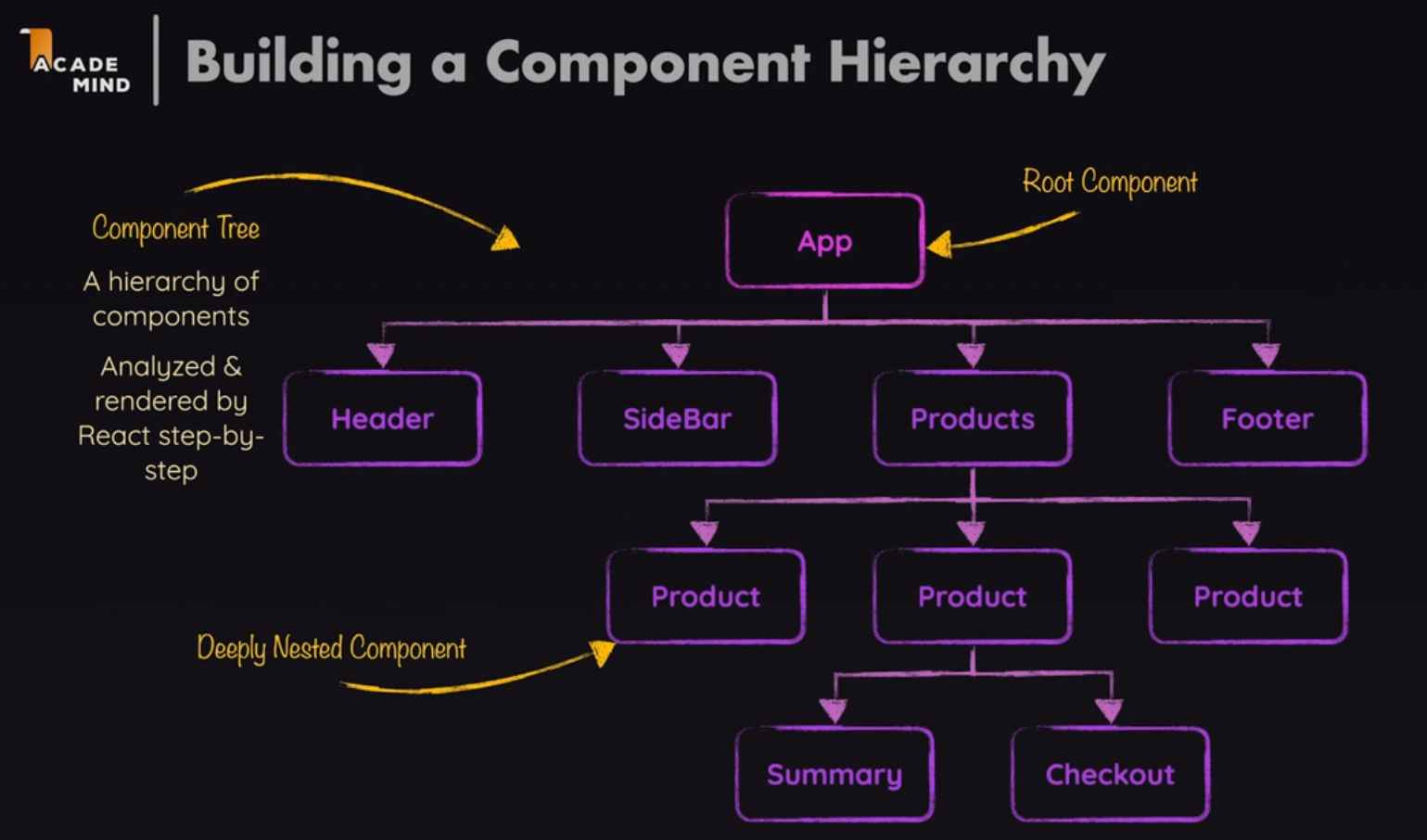
Graphical user interface, diagram

Description automatically generated

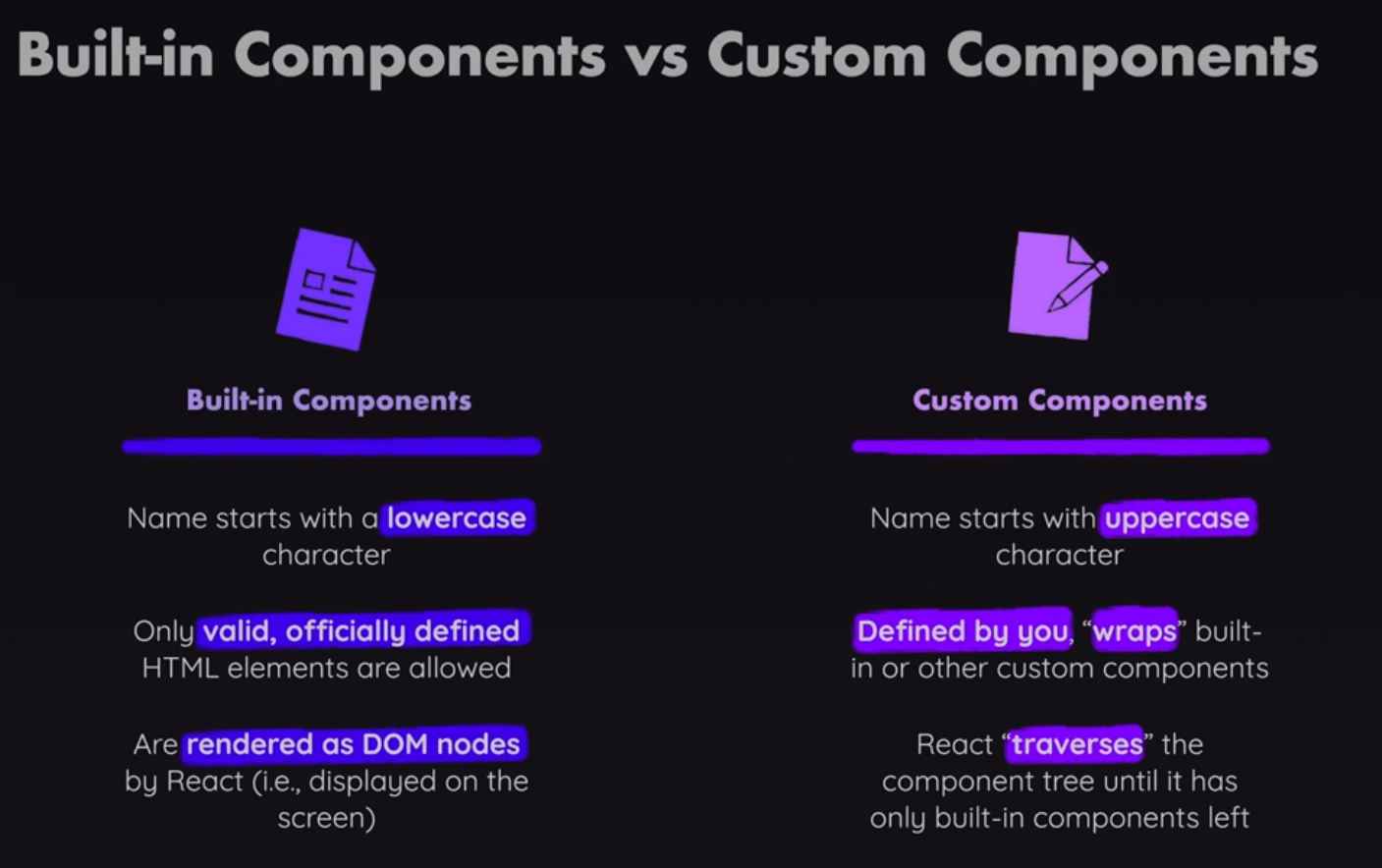
**createRoot & render function:** responsible for rendering a single root component which then in turn may contain as many nested components as needed.



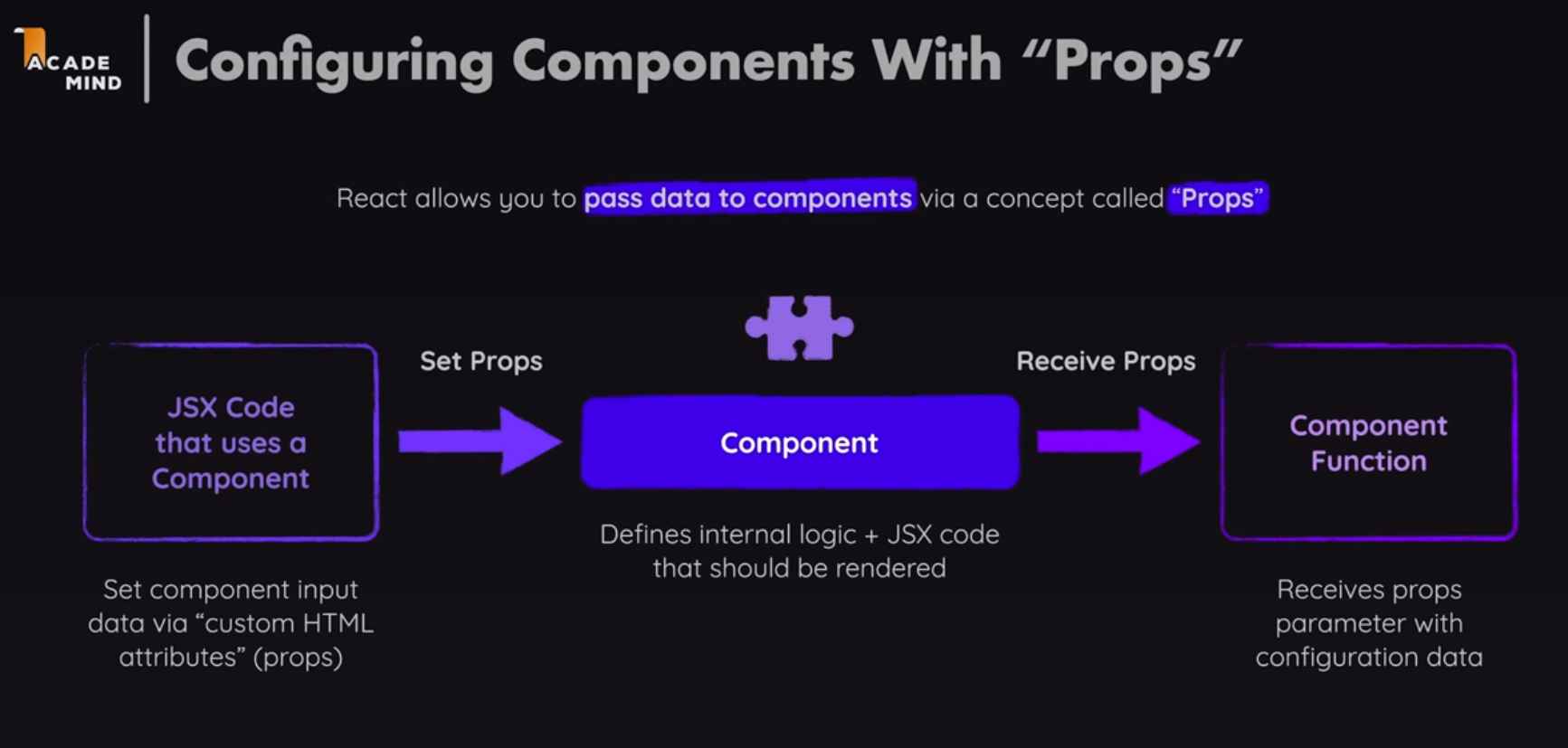
Ultimately, we end up with a component hierarchy



Built in components like header, image, div are rendered as DOM nodes in React. On the other hand, custom components are just functions and executed as functions by React.

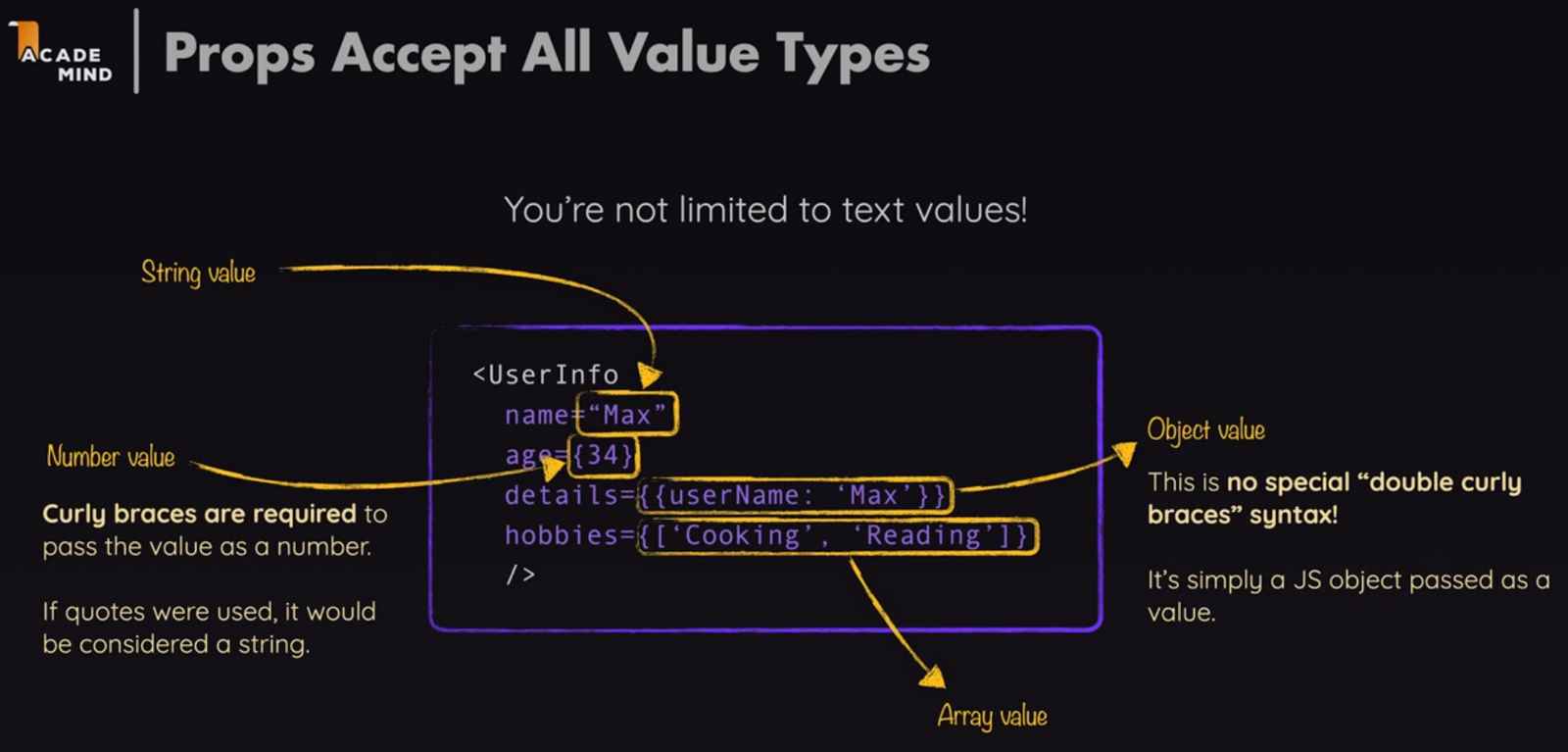


**Props in React:** let’s say we have custom component which should output a list of items where the goalItem should be output dynamically. We can pass data to custom component by adding a attribute. And inside that component, we can then get access to all these attributes.





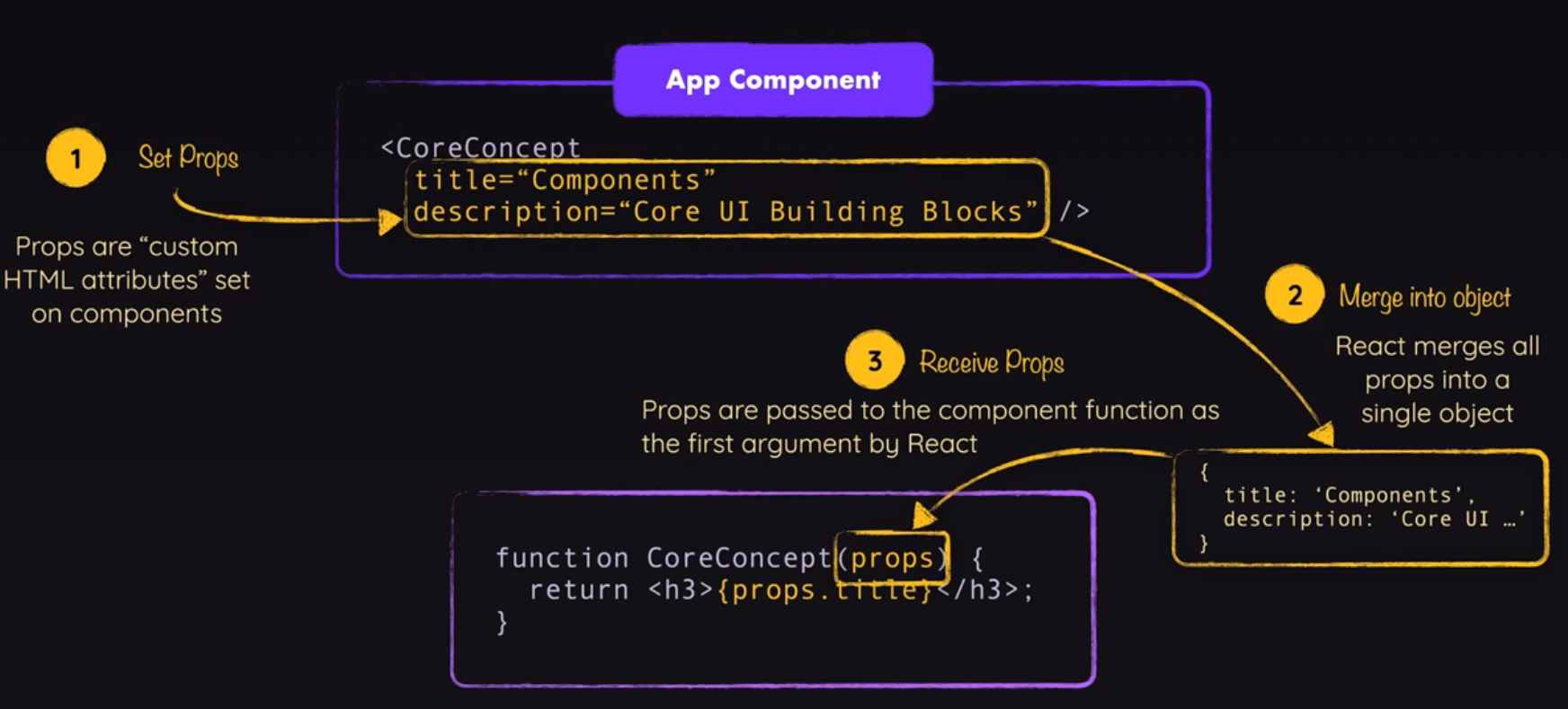
You can pass all kinds of values for your props



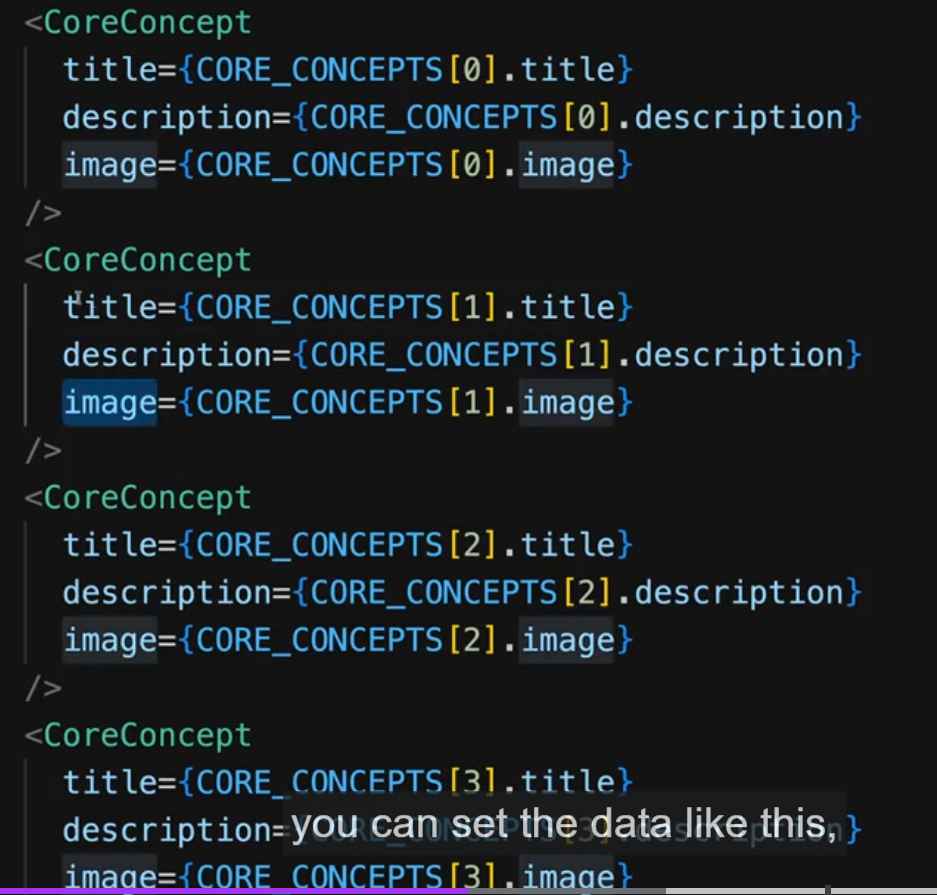
However, when pass parameter to the function, it only allows 1 parameter as an object



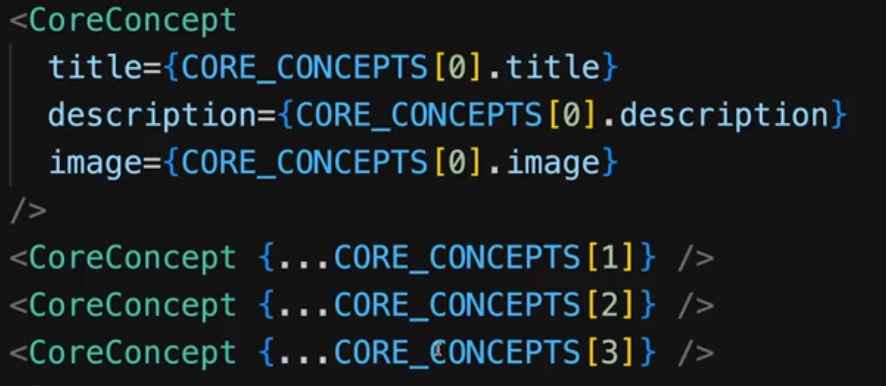
That “props” parameter is an object.



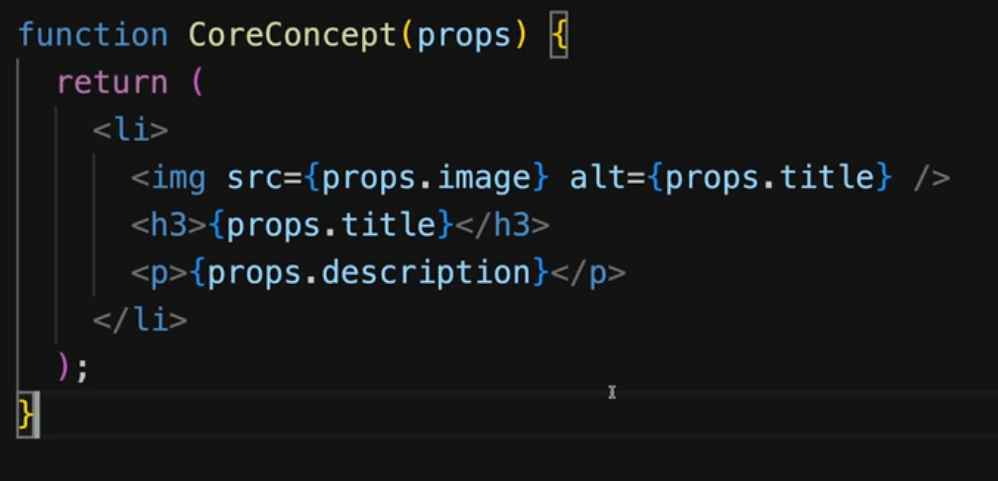
Example of how props are used with CORE\_CONCEPTS imported externally



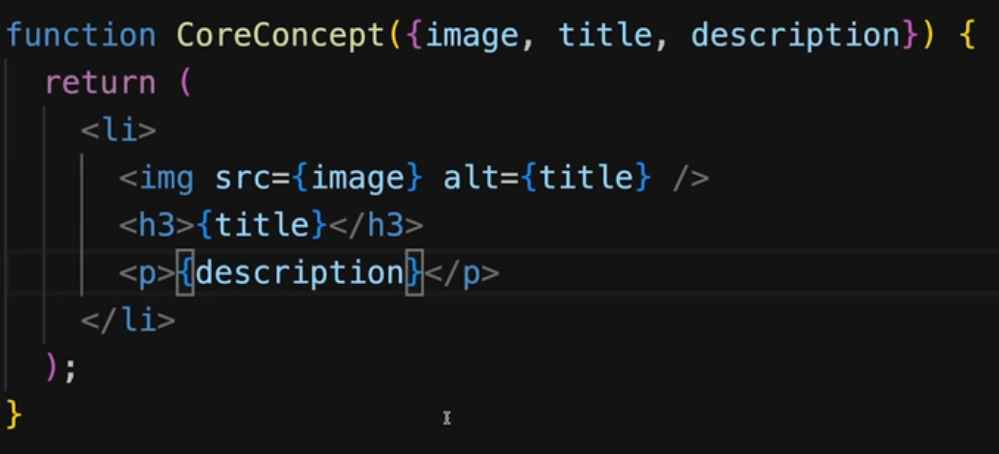
However, it could also be written like this



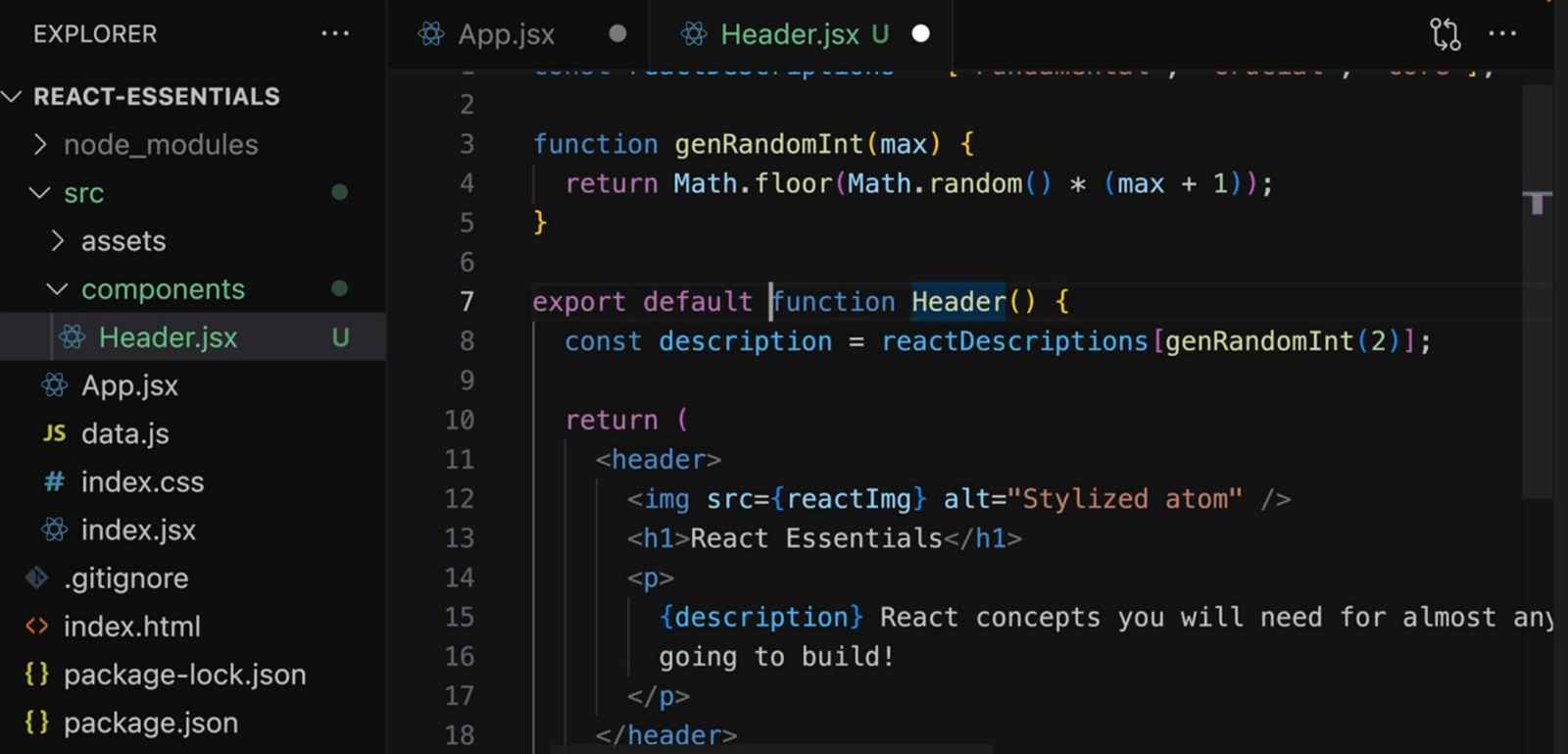
Also, instead of defining “props” parameter, we could also deconstruct the “props” object



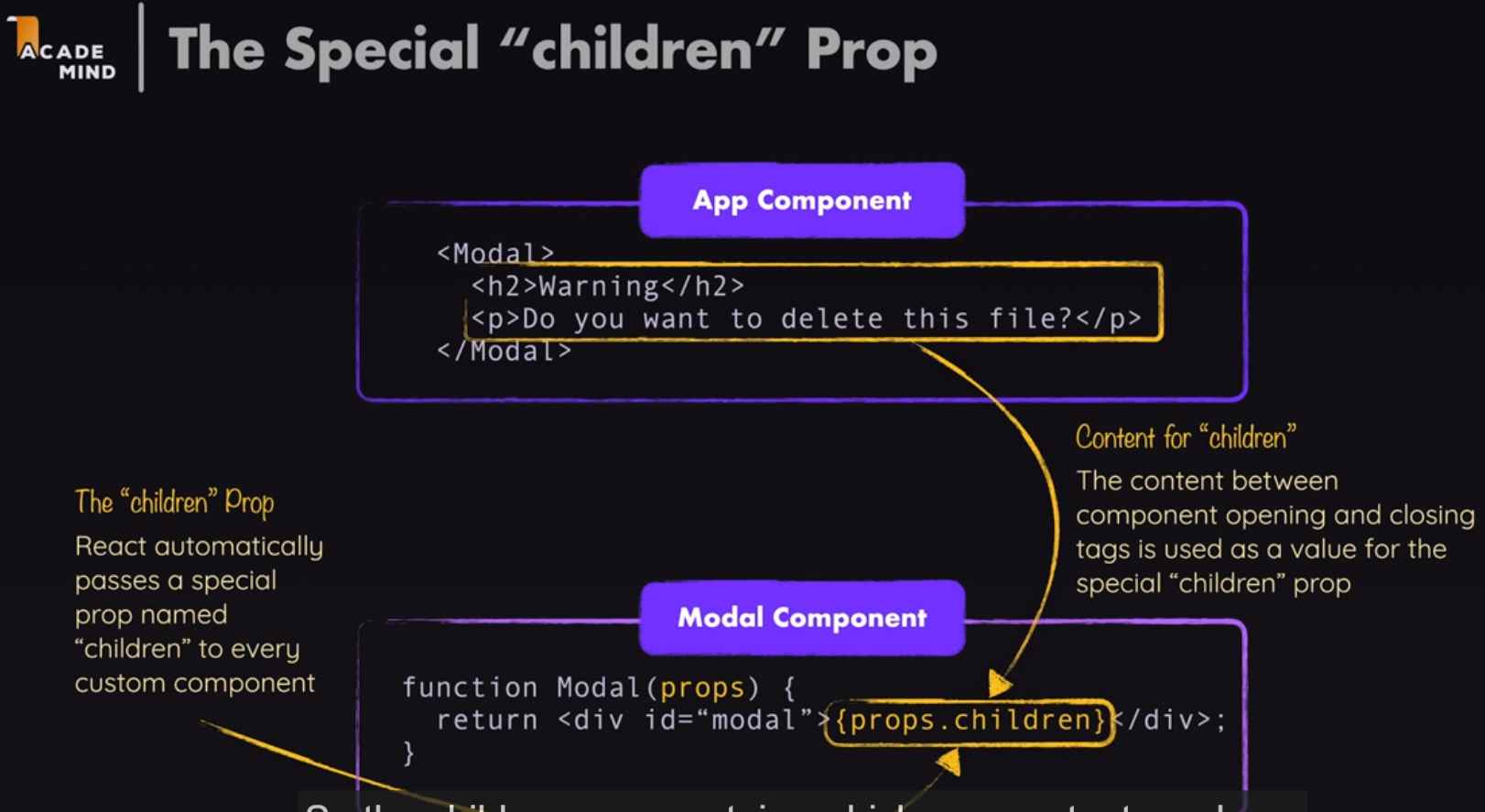
Turn it into this

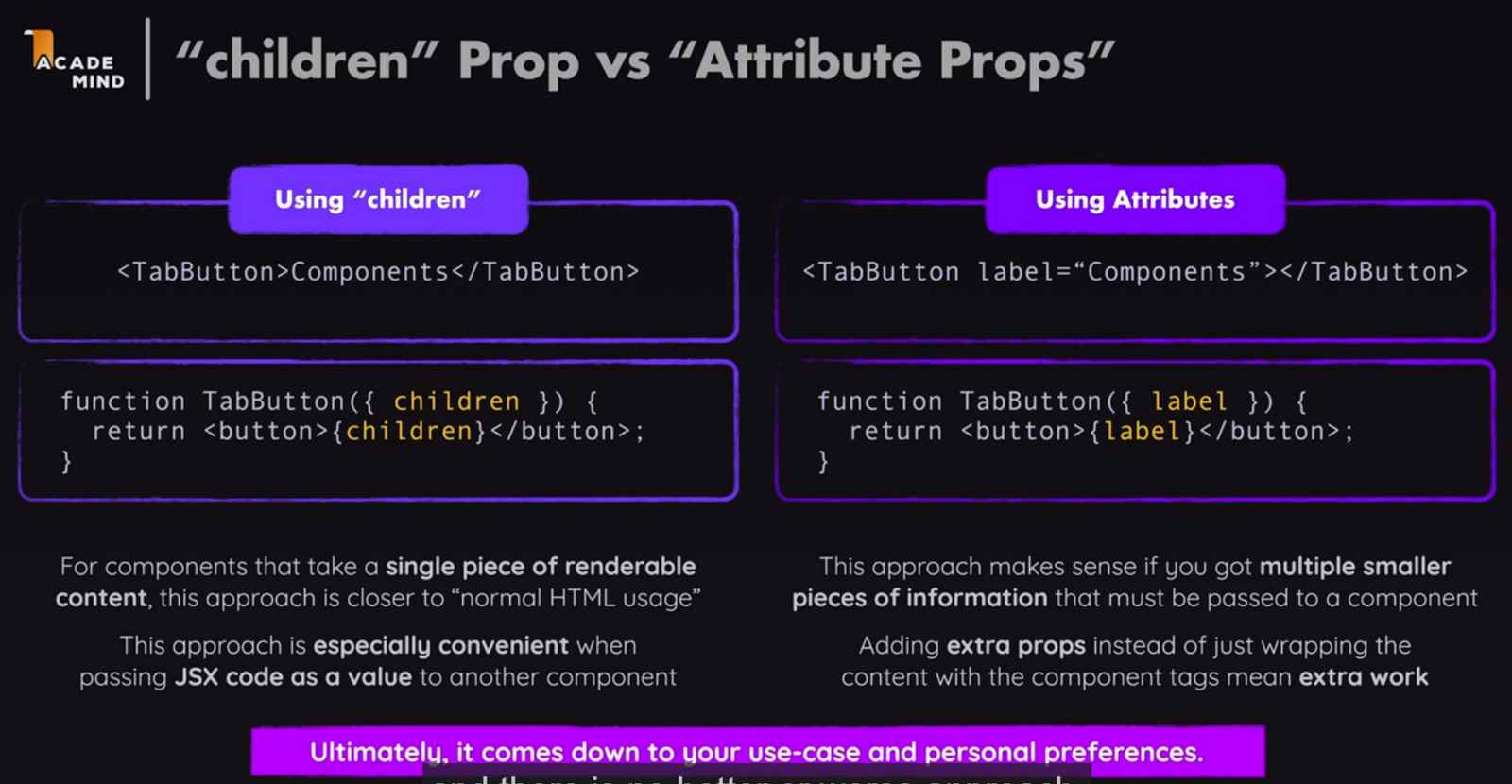


Best practice is to minimize the size of components and move to the same file name JSX with the component name.



The children props contain whichever content you have between your component text, it doesn’t appear with the help of “children” attribute (we don’t need to have something like <Modal children =””> for it to appear in the function), it could be a text or any complicated elements.





In order to create event in React, we use special attribute, for example: “onClick” and the value would be a function. The function must NOT be executed or must not add paranthesis (not handleClick() but only handleClick). We are using function as value.

