There’s more than one way to build something in React ⚛️

Here’s a guide though👇

1. Break the UI into a component hierarchy

2. Build a static version

3. Identify the complete representation of the UI state

4. Identify where your state should live

5. Add inverse data flow

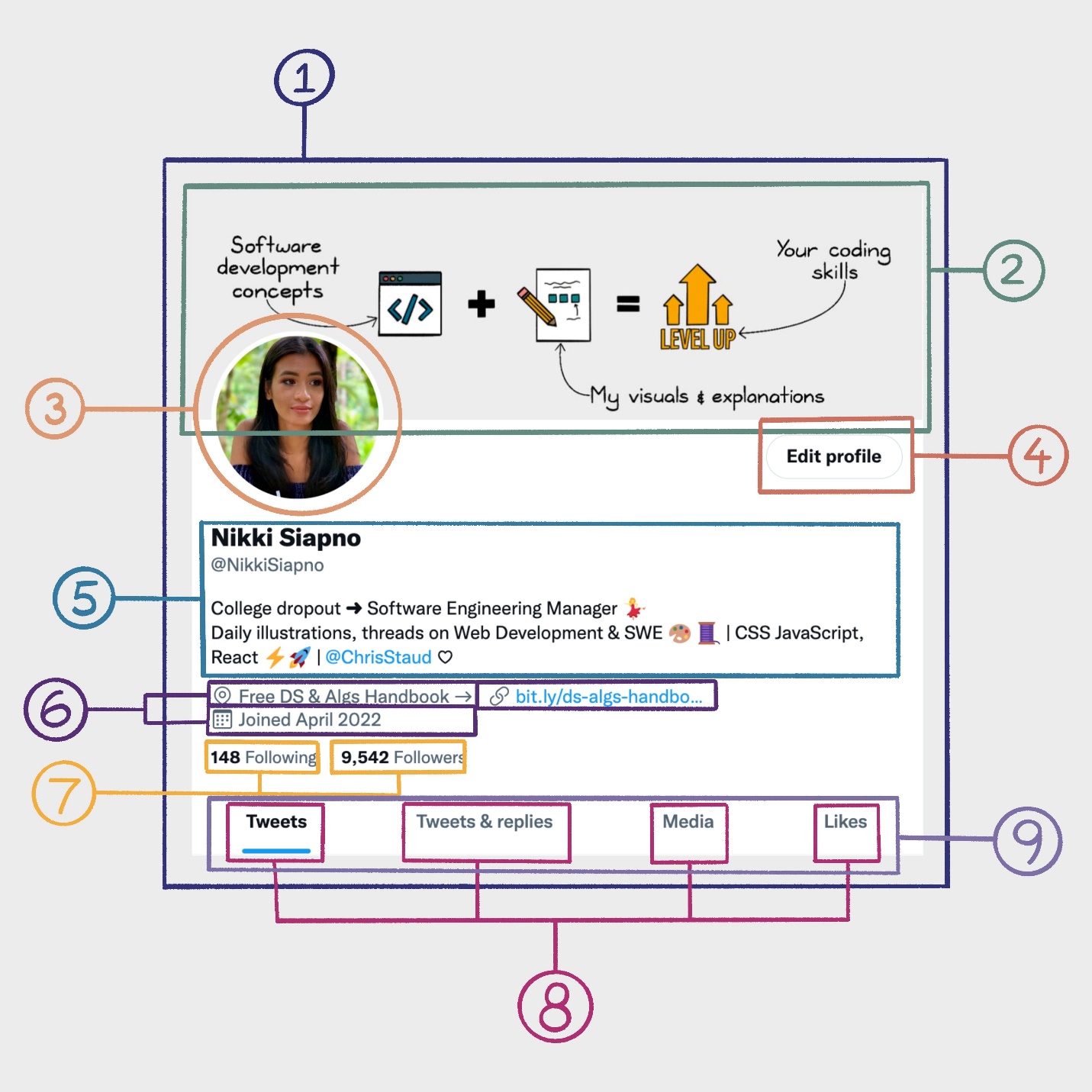
1) 🌳 Break the UI into a component hierarchy 🌳

Start with the end in mind.

Identify what the end is meant to look like so that you can break down what components make up the UI 💡

Having/creating a mockup makes this easy 🎨

Draw boxes around each component & name them.



1a) How do you know what should be its own component? 🤔

The same way you should decide whether to create a new function or object.

The single responsibility principle - SRP 💡

A component should only do one thing ✅

If it grows, it should be decomposed into subcomponents.

2) 📄 Build a static version 📄

You can build bottom-up or top-down ▲▼

In small apps it’s easier to use the top-down approach, ie; start at the top of the hierarchy.

In large projects it’s easier to build bottom-up and write tests as you build.

2a) To build a static version of an app that renders a data model;

You should build components that reuse other components, & pass data via props. ✅

State should not be used to build a static version ❌

State is reserved only for interactivity - data that changes over time.

2b) React docs suggest this decoupling approach to building (separating static & interactive).

They say it's a more efficient approach.

IMO, I use both approaches depending on what I’m building.

I suggest learning both techniques & apply what works best for the situation.

3) Identify the minimal (but complete) representation of the UI state

To make UI interactive, you need to be able to trigger changes to the underlying data model.

React achieves this with state 🙌💡

3a) To build your app cleanly, keep state DRY ✅

Think about a minimal set of mutable state that your app needs ✨

For example, for a To Do List App, keep an array of the TODO items (state). Don’t keep a state variable for the count of items, just take the length from state 💡.



4) 🏡 Identify where your state should live 🏡

After identifying the minimal set of an application’s state, you need to identify which component(s) mutates, or owns state 🤔

4a) React is all about one-way data flow down the component hierarchy 💡

Identify every component that renders something based on a set of state.

Find a common owner component - a single component above all the components that need the state in the hierarchy.

4b) Either the common owner or another component higher up should own the state ✅

You can then pass it down via props to all the components that also rely on the state.

If you can’t find a component where it makes sense to hold state, create a new component for holding state.

5) 🙃 Add inverse data flow 🙃

Step 4 allows apps to render correctly as a function of props and state flowing down the component hierarchy.

Data often also needs to flow the other way; components deep in the hierarchy need to update state in components higher up 🌟

5a) To do this we pass setState() from the higher-up components down the hierarchy 💡

That way, components deeper in the tree can call setState() when required (like a form input event) to update the state living in the common owner component (a form for example) ✅

That's a wrap!

If you enjoyed this thread, don't forget to like, comment, and retweet the first tweet!

I create threads, and hand-drawn illustrations to level up your software development game 🧵 🎨 🚀.