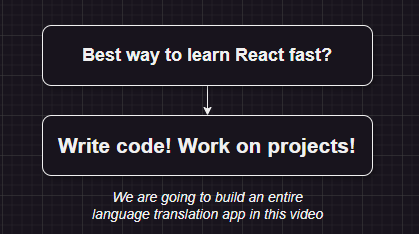
**React.js**

**General**

**Use this to recap JS:** [**https://github.com/Asabeneh/30-Days-Of-JavaScript/blob/master/readMe.md**](https://github.com/Asabeneh/30-Days-Of-JavaScript/blob/master/readMe.md)

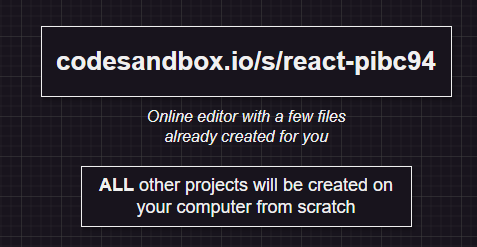
**Section 1: Let’s Dive in the world of React**

****

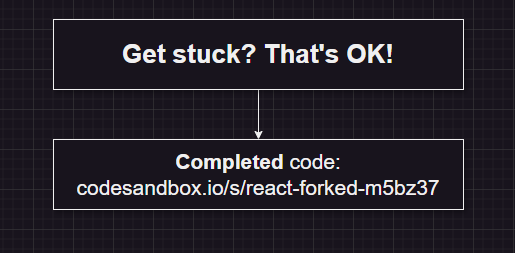
**We are going to Learn React but first we are going to get introduced with an already created React app, which is a Small Text Translator web app.**

****

**Approach of this app is we type a text in the Text Area then we select the language whose translation we want to see. Translated text will be shown in output area.**

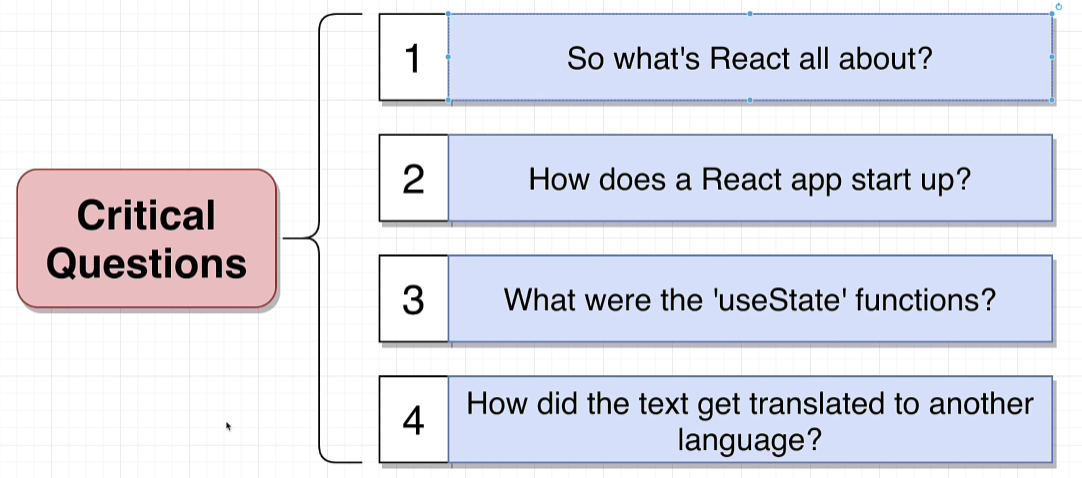
****

**You will find the sandbox of the code of this translation app.**

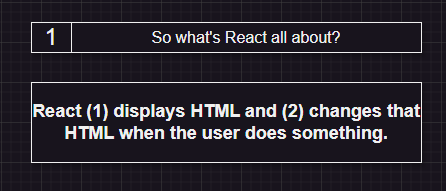
****

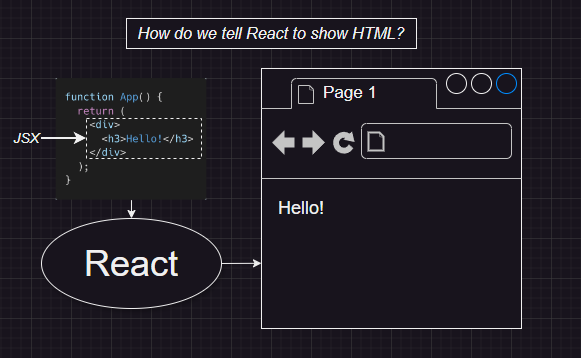
**Above image contains the info of complete code for this Translation App.**

**Now introduce the component that are already created into the App.js and pass states which contain user input TEXT and default language that will be shown when the screen will load.**

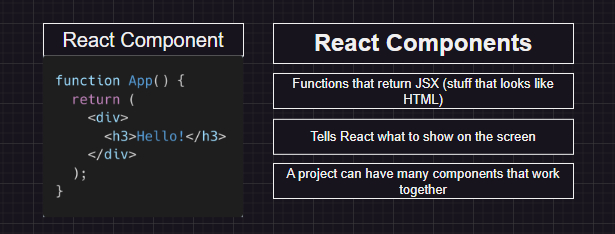
****

**Some Important Questions to go through in order to understand the translator code.**

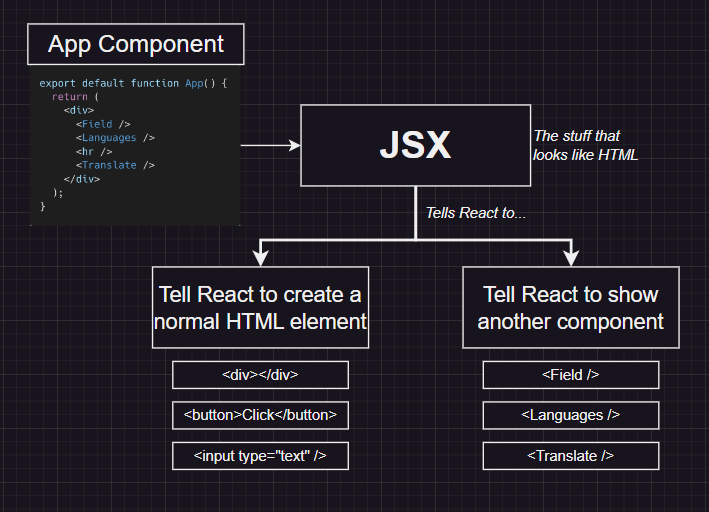
****

****

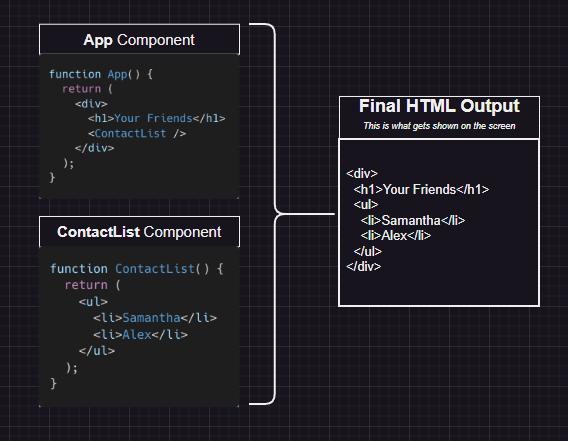
**JSX tell react to convert the JSX code to HTML Code and start working on that.**

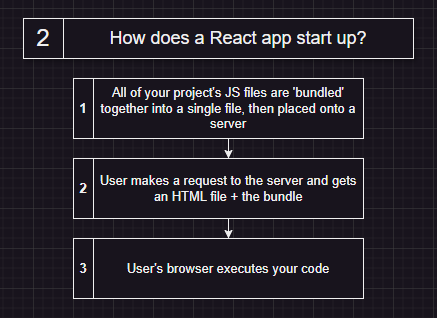
****

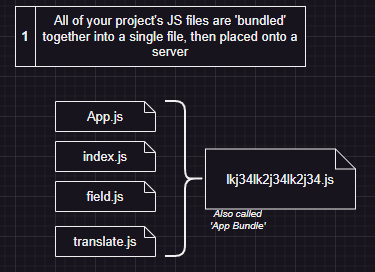
****

****

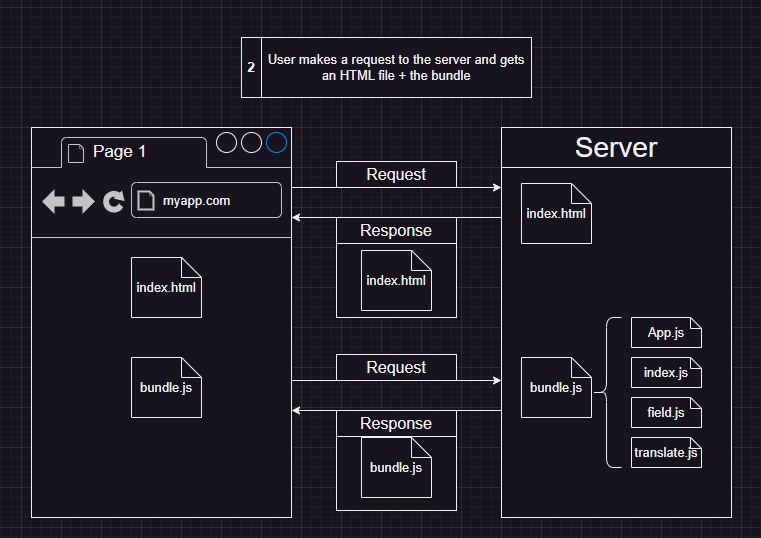
**How JSX tell reacts what to show HTML or another component and then collaborate every code to finally show in HTML (index.html) file under the root div.**

****

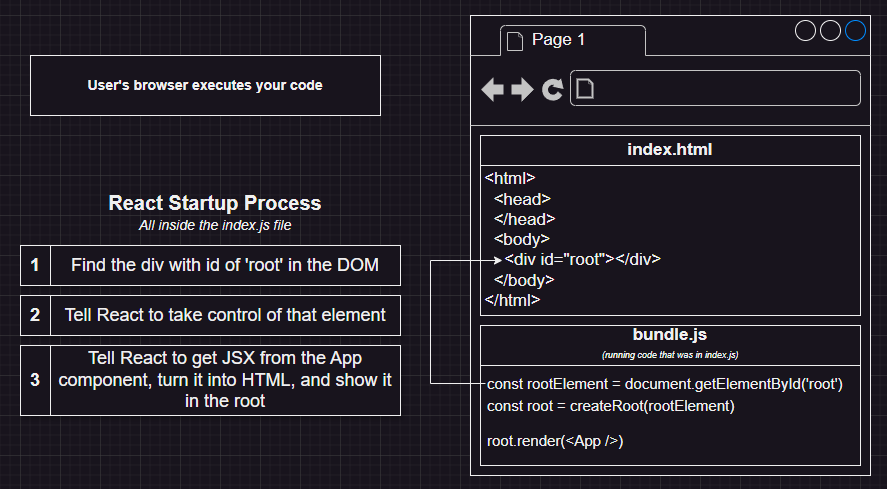
****

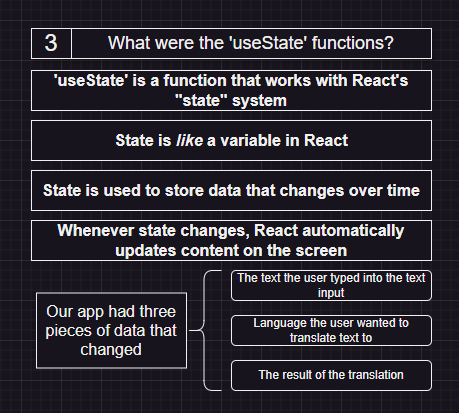
****

**When client will go to an URL, this trigger a call form to server which fetches the index.html file then your HTML will send another request to fetch the files that are linked with your html with <link>, <script> and more tag.**

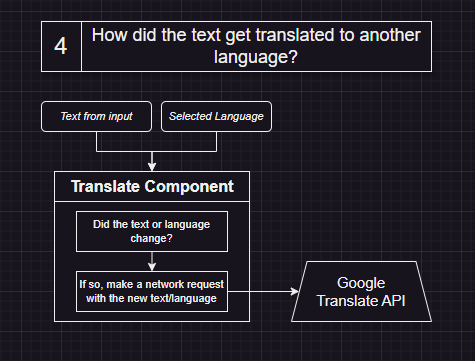
****

**The rootElement will get the reference of the div#root present in index.html then createRoot(rootElement) will establish a connection that what next(root.render(<App/>)) is executed will display under the root container.**

****

****

**useState() will keep record of those variable that may or may not change in user interaction. The variable that is declared with help of used state have the ability to keep track of the interaction and changes occur to it. In this case Text and Language are the state variable.**

****

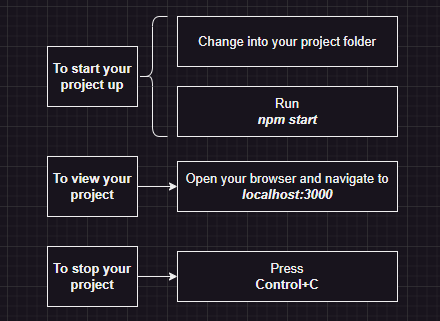
**As Text and Language are States Variable they and when they change React will know that the values is changed they trigger a request to the GOOGLE API for translation of TEXT into what user chose as language. API will do its JOB and then the converted result is printed in output.**

**MY LINK FOR THE SANDBOX Translation app Code:** [**https://codesandbox.io/s/react-forked-k4crth?file=/src/App.js**](https://codesandbox.io/s/react-forked-k4crth?file=/src/App.js)

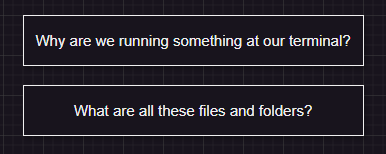
**Start a new react project: npx create-react-app <project-name>**

**This is old way to make react app and now days it’s not preferable because it download multiple dependencies and install packages before giving the ready app to you. Its is said to be outdated because some of the dependencies that was preinstalled is not gonna be used by you it is a waste to space + when we create app with this method we see longer setup time as it internally download and installs packages for you. In New React doc it is suggested to use remix OR Vue as it just prepare the folder structure but does not installs for you make the project more lite and less memory.**

**In official Doc the most preferred way is to use Frameworks like Next.js as to contain a lot of features and still have all react property supported.**

****

**NPM = Node Package Manager, NPX = Node Package Extractor.**

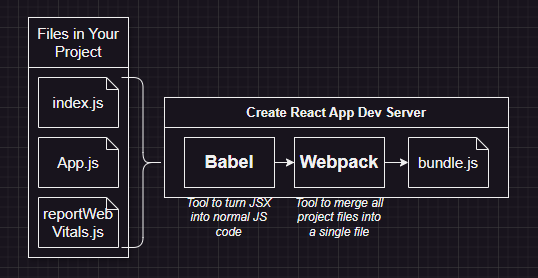
****

**Browser does not know how to execute JSX(not value JS Code used because of convenience).**

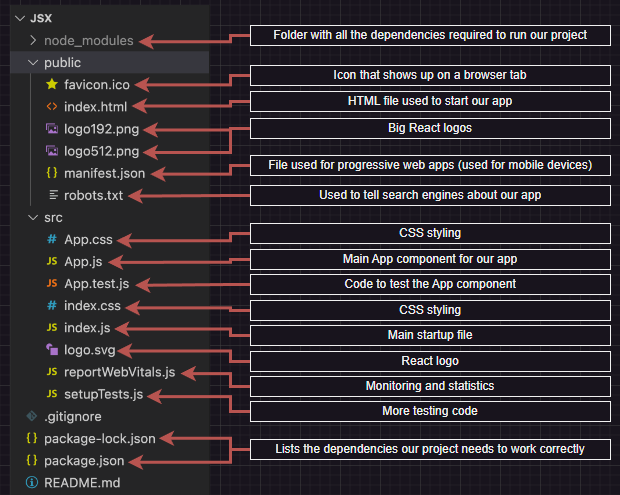
****

**Process that convert the React JSX code to browser compatible code so that it can display.**

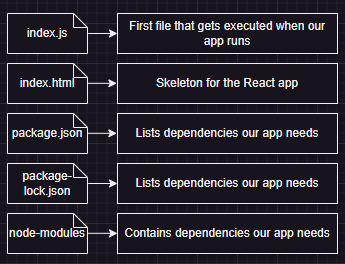
**The CRA dev server converts the code u write in react to JS compatible code combines the code and make the bundle ready to be read by browser and get executed.**

****

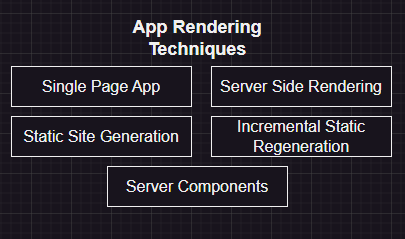
**Most of the files we get while installing react are not necessary to run the actual app. It’s just some extra things that make the app looks or behave well.**

****

**Important Five:**

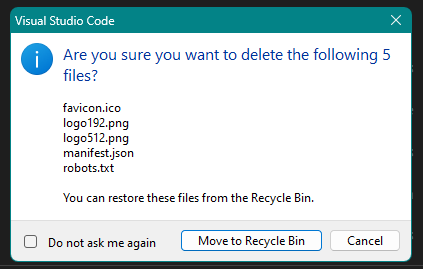
****

**Read and explain below image:**

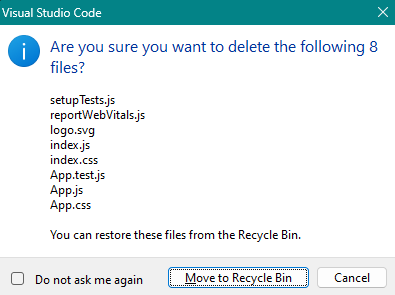
****

**Section 2: Creating Content with JSX**

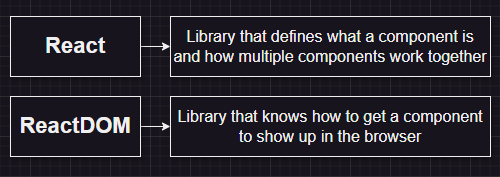
**Now we delete all the unnecessary files from the folder.**

****

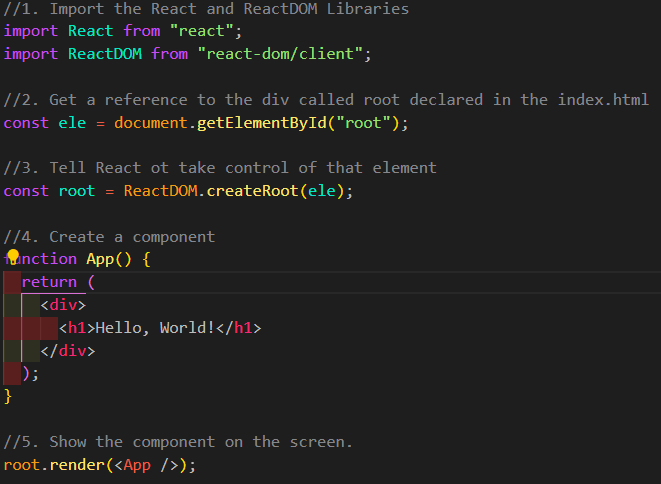
**Then delete all the files form src folder and create your own index.js file and code its content**

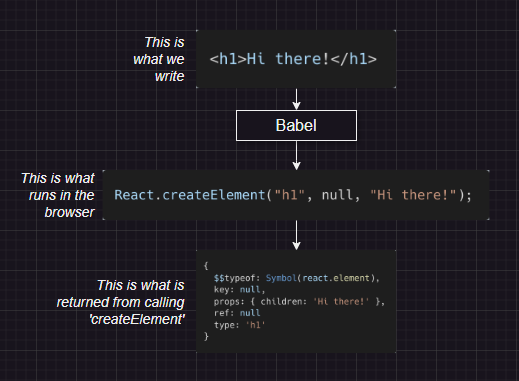
****

**After deleting all the extra files from the react folder (src and public). Make a file called index.js in src folder and configure it.**

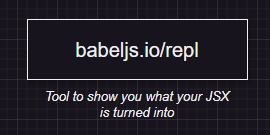
****

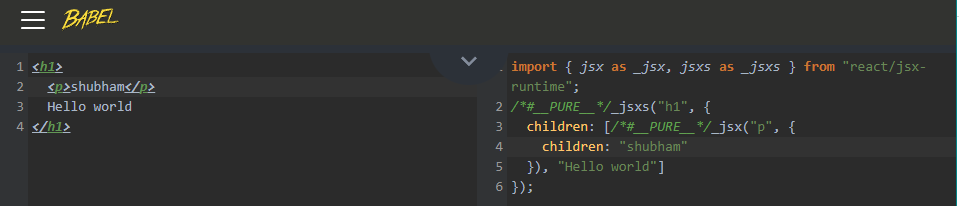
**React in not self-sufficient to make a web react app. ReactDOM take the component form react -> converts them into html and then display it on the browser. React library in itself does not communicate with browser as react have feature for both web and native mobile applications.**

****

****

**An online editor to show the conversion of JSX into JS code done by Babel**

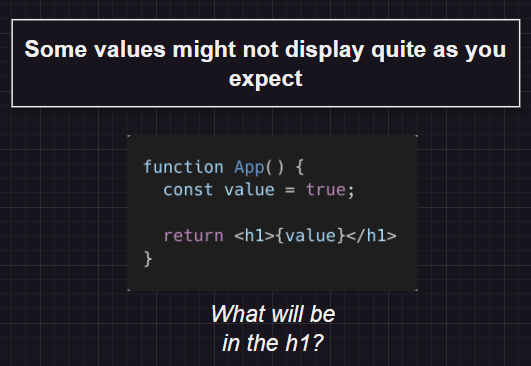
****

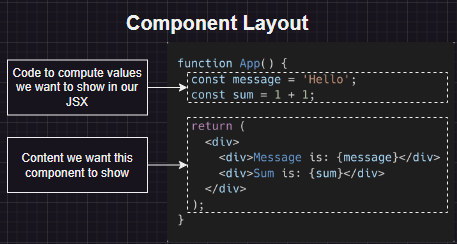
**Example:** ****

Conclusion : using JSX make the react dev life easier. JSX will never show any thing else inside it if its not returned.



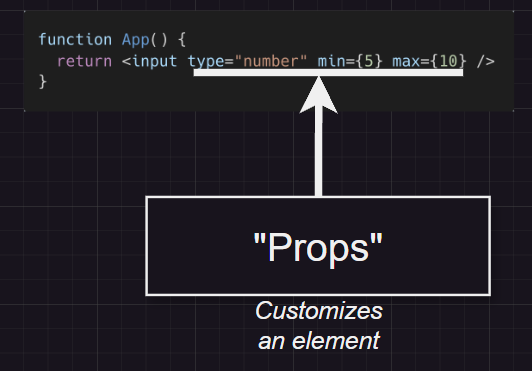
React will only display string and number (when we try to print a variable) not boolean (true or false), undefined, null and etc. when we try to print an object{} it will throw an error on console.

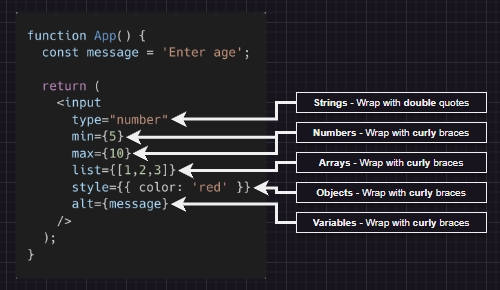




Declaring variable is not necessary but it’s a good practice<div>{new Date().toLocaleTimeString()}</div> works just fine but not a good practice.

**In html we have attributes for tags, in react we have props**

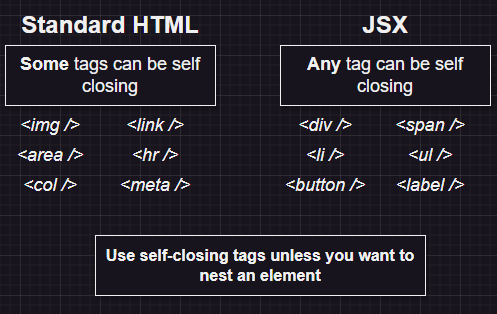
****

****

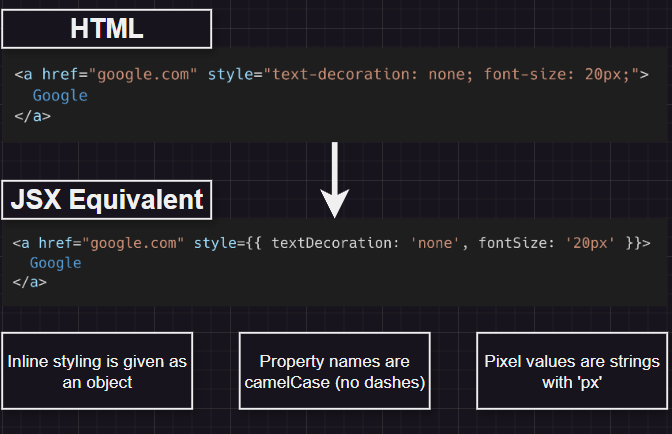
**Difference between props and attributes (GOOGLE and ADD)**

**As we know we cannot use object to print on web but we can use onject to pass into a prop, just like any other value.**

**SOME KEY DIFFERENCE BETWEEN HTML AND REACT**

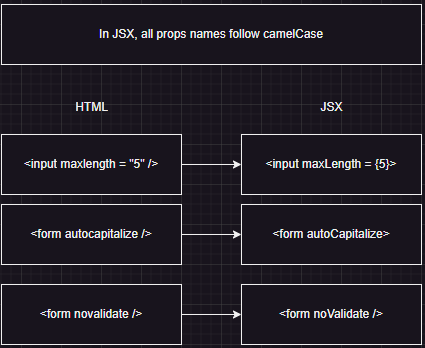
****

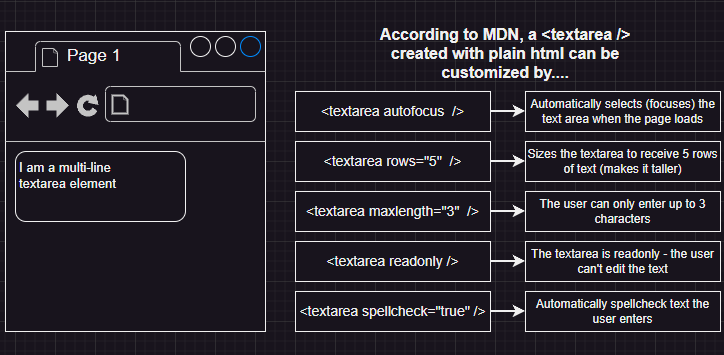
**The way we write inline Styles in React is different what we are used to in HTML.**

****

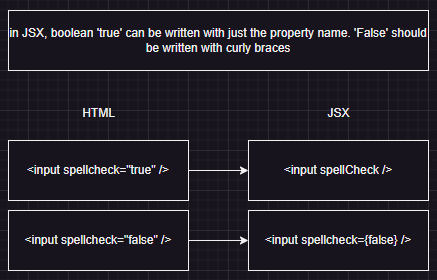
**Converting HTML into JSX require some steps:**

1. **All props names follow camelCase, and when we ran into an error, check web console.**

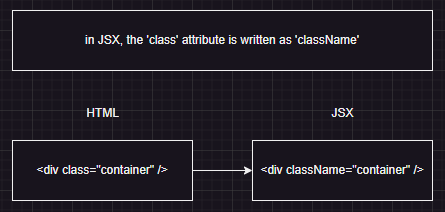
****

****

1. **Number attributes use curly braces.**
2. **Boolean ‘true’ can be written with just the property name. ‘False’ should be written with curly braces.**

****

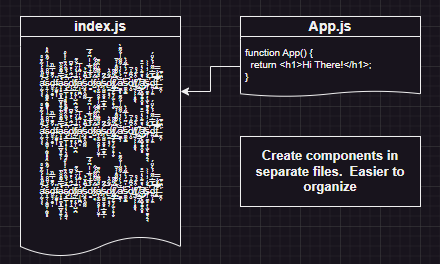
1. **The ‘class’ attributes is written as ‘className’.**

****

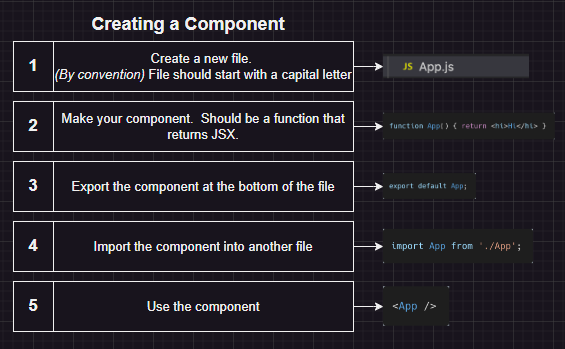
1. **In-line styles are provided as objects styles= {{color: ‘red’}}.**

**GIVE LIVE DEMO**

**Writing everything into one file is not good**

****

**In previous step we declared the App component inside index.js but now we will separate this into another file name as App.js containing App component and then import it in index.js**

****

**CHEETSHEET FOR JSX:** [**https://jsx-notes.vercel.app/**](https://jsx-notes.vercel.app/)

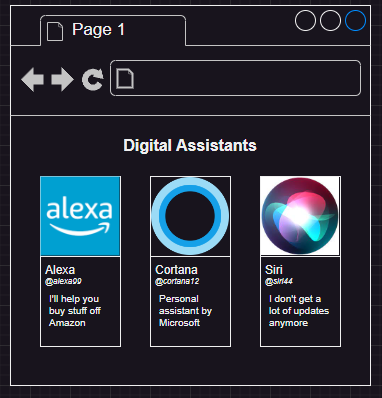
**Section 3: Building Reusable component**

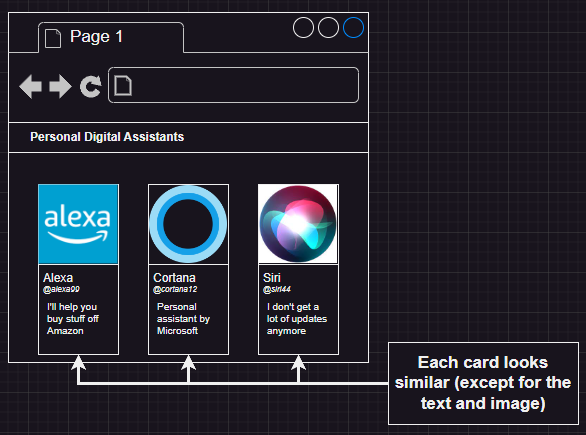
**Starting with react-app-2 with vite+react**

The reason Vite requires .jsx extension for JSX processing is because in most cases plain .js files shouldn't need full AST transforms to work in the browser. Allowing JSX in .js files means every served file must be full-AST-processed just in case it contains JSX.

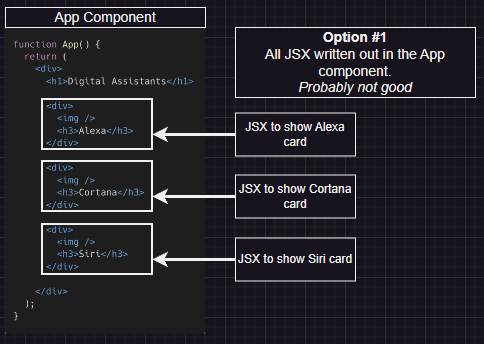
Vite works only when I kept the index.html in main project folder but like CRA when index.html is put in public folder it just do not work.

**New Project to get hands on making components for similar looking cards:**

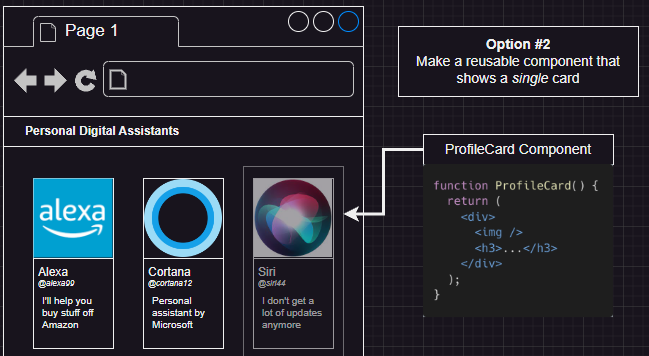
****

****

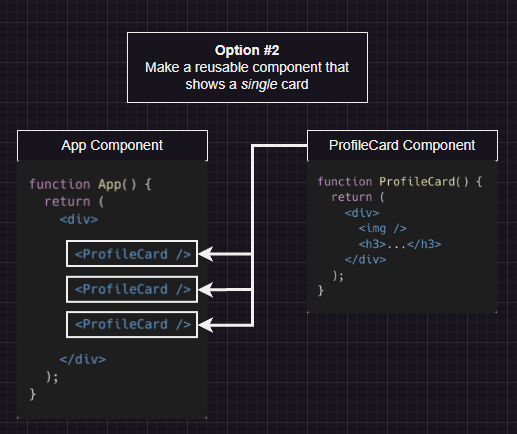
**Approach 1: Duplicated JSX**

****

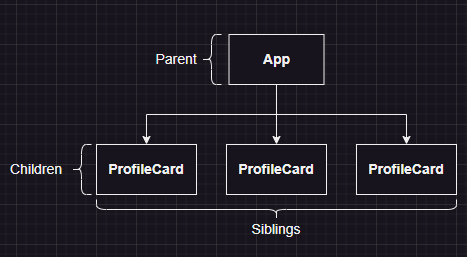
**Approach 2: Make reusable component**

****

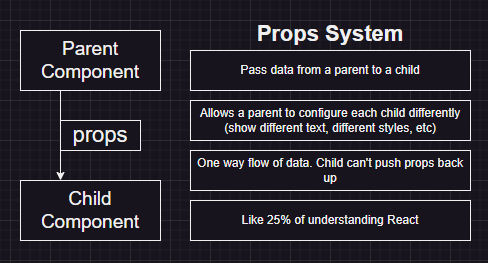
**Use component multiple times with dedicated inputs:**

****

**Component Hierarchy:**

****

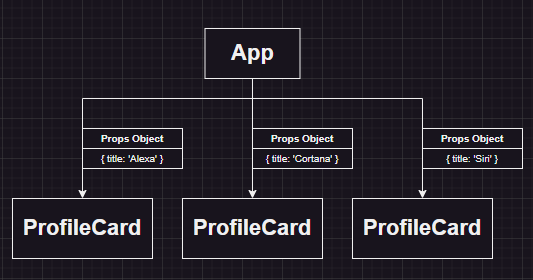
**Props is called from top to bottom means from parent to child, but not other way around.**

****

**Our Approach the build:**

**Now we need props and want to send form parent to child component, no other way back.**

**It’s a only one way flow of data. In this code App.jsx is parent and ProfileCard.jsx is Child so, each time when we send the data from app to profilecard we send it as attribute and receive as props (props.<NameOfThePassedAttributeFromApp.jsx>). This simplifies the code and make the inheritance of data form parent very concise and easy to use.**

****

**Now here is how we approach to the code, the steps in boxes is broken down into steps of code. After mentioning the attributes in parent, react will collect those details and treat as an object to be used by child (called props).**

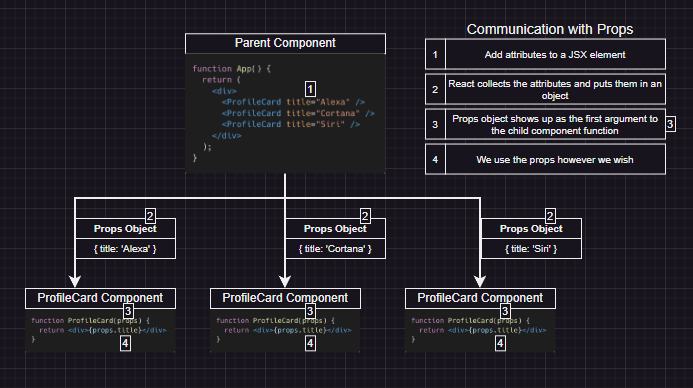
**As we know React treat the data which is sent from parent as object, the concept of destructing can be applied to as,**

**Code1: const {title, handle} = props.**

**Code2: const title = props.title, const handle = props.handle**

**Code1 == Code2**

**Make sure to use the same name as the name being passed from parent else it will unable to recognize and will throw an error as it’s unable to tell which prop value we are refereeing too.**

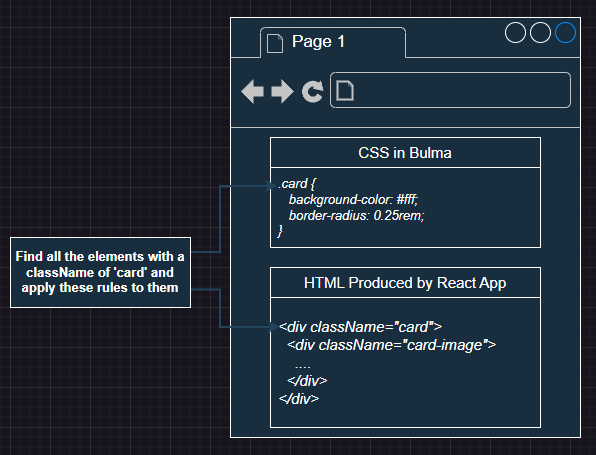
****

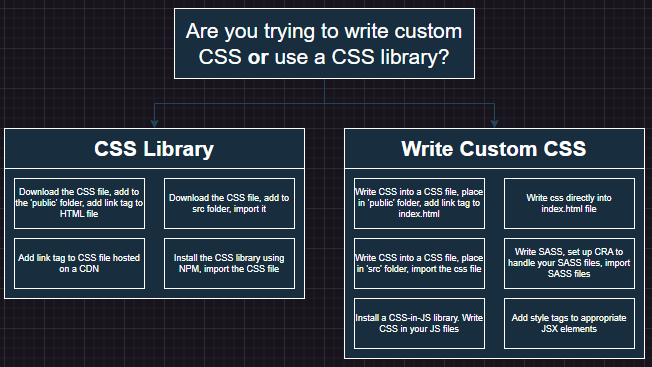
**Install Extension: React Developer tool (For me tools are not showing up in browser)**

**Now we include image into the project:**

**Steps:**

1. **Make a folder an add pictures of Alexa, Cortana and Siri.**
2. **Import into App.jsx and send it as props to ProfileCard.jsx.**
3. **Use to props to show in browser.**
4. **Now, it’s time for some CSS. We are going to use bulma.io to make things look good.**
5. ** You install the CSS library use the CSS classes as said in the documentation of the library and get what was shown in the website as reference to the classes look.**

****

****

****

1. **How to use BULMA.**
   1. **npm install bulma**
   2. **Then import "bulma/css/bulma.css" = correct way to import a folder which is in node modules folder. Using import "../bulma/css/bulma.css"; will cause error as I will search in the main project folder not in node modules (which is in main project folder).**
   3. **Make change in profileCard.jsx code and apply CSS className form bulma, what classNames is used is decided by bulma.io.**
   4. **What class and div structure is need to be used is shown in the code page where you found what card format is good for your code. Follow that.**
   5. **Now the alignment of the card is not that good so. Addgo the columns tab and choose what column count u want and follow the className and div structure to get the result and good looking cards.**
   6. **Bulma documentation -> layout -> hero -> choose the way you want your header to look like.**

**Completed**

**Section 4: State: How to change your app**

**For this section I am going to use PARCEL to make react for application.**

**Conclusion you cannot make react app with PARCEL: it makes only web app no react I guess or there is way but I read only starting of the documentation page so, will update this later.**

**npm create vite@latest it is.**

**But with a change:**

**√ Project name: ... react-app-3**

**√ Select a framework: » React**

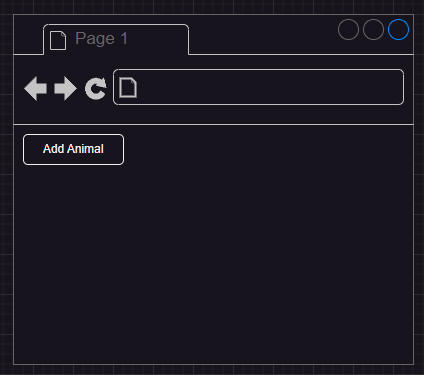
**√ Select a variant: » JavaScript + SWC**

**SWC is a free and open-source JavaScript transcompiler like Babel, but unlike Babel it can significantly speed up build and development time due to SWC's fast conversion capabilities, however it may not support all Babel plugins. SWC plays an integral role in the Vite ecosystem.**

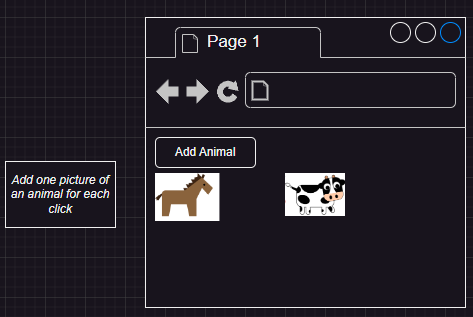
**So let’s see what challenges I face for not using BABEL as compiler.**

**I Vite its import to import React, without it it will not display the pages. Import even if you are not using it.**

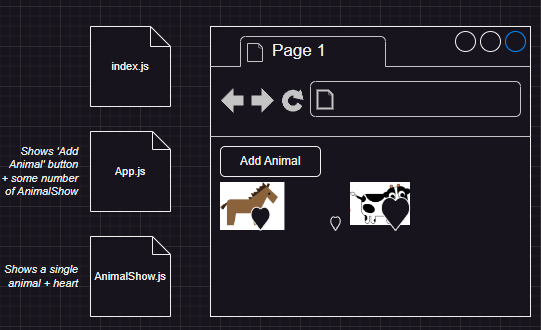
**App Description:**

****

**Show a page with no animal.**

****

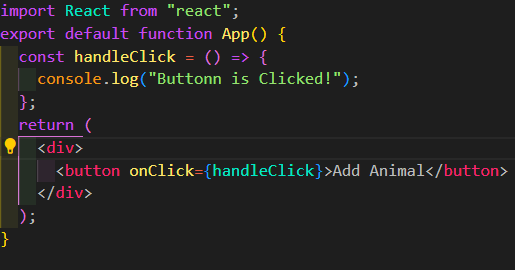
**As soon as add animal button is clicked, add animal with a heart logo on bottom right.**

****

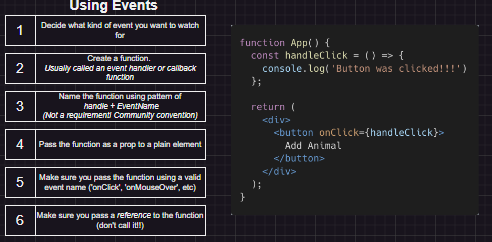
**When number of times animal image is clicked, keep growing the heart logo size.**

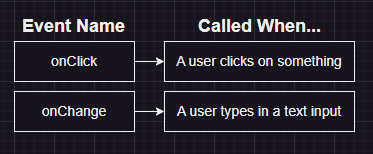
**In this upcoming project we will learn about events.**

**Show the basic on how to use onClick in React**

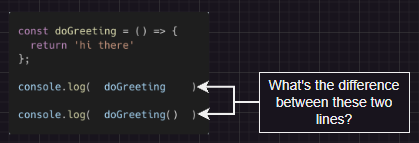
****

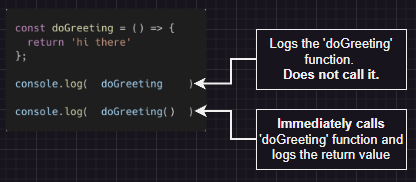
****

****

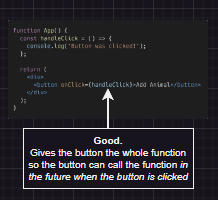
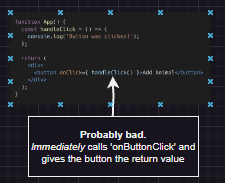
****

**Go to react official website to show what number of event we can use:** [**https://legacy.reactjs.org/docs/events.html**](https://legacy.reactjs.org/docs/events.html)

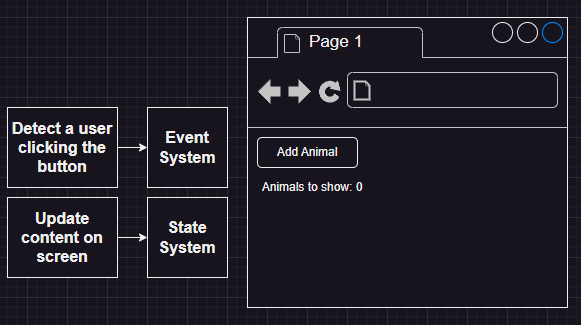
**What this code snippet do? **

**Answer: **

**Why we do not use handleClick() an use handleClick without ()**

** **

**Introduction of State System (State management into react)**

****

**We will take a small example to get know how states work. The example we going to take can be achieved with event handler using vanilla JS but in react same thing can be done with states, that is re-render the web app but no reloading.**

**VDOM vs DOM**

**DOM (Document Object Model)**

**Definition: The DOM is a programming interface for web documents. It represents the structure of a document as a tree of nodes, with each node corresponding to an element or a piece of content in the document (such as an element, attribute, or text).**

**Manipulation: When you manipulate the DOM directly (e.g., using JavaScript), it updates the actual HTML elements on the web page. This can be slow and inefficient, especially when many elements are being updated frequently.**

**Performance: Direct manipulation of the DOM can lead to performance issues because each change triggers reflow and repaint processes in the browser, which can be expensive operations.**

**Virtual DOM (VDOM)**

**Definition: The Virtual DOM is a concept implemented by libraries like React. It is a lightweight copy of the actual DOM, represented as a JavaScript object.**

**Manipulation: When changes are made in a React application, they are first applied to the Virtual DOM. React then calculates the difference (or "diff") between the previous and the current Virtual DOM state.**

**Efficiency: React updates only the parts of the actual DOM that have changed, based on the diff calculation. This minimizes the number of direct manipulations to the actual DOM, leading to better performance.**

**Reconciliation: The process of updating the actual DOM based on changes in the Virtual DOM is called reconciliation. React efficiently updates the DOM in a way that reduces the performance cost associated with direct DOM manipulations.**

**Comparison**

**Direct Updates:**

**DOM: Directly manipulated by JavaScript, leading to potential performance bottlenecks.**

**VDOM: Changes are first made to the Virtual DOM, and then only necessary updates are applied to the actual DOM.**

**Performance:**

**DOM: Less efficient for frequent updates or large-scale changes.**

**VDOM: More efficient, as it minimizes direct DOM manipulations and optimizes the update process.**

**Complexity:**

**DOM: Simpler in concept but can become cumbersome and inefficient with large, dynamic applications.**

**VDOM: Adds an abstraction layer that simplifies state management and improves performance in large, dynamic applications.**

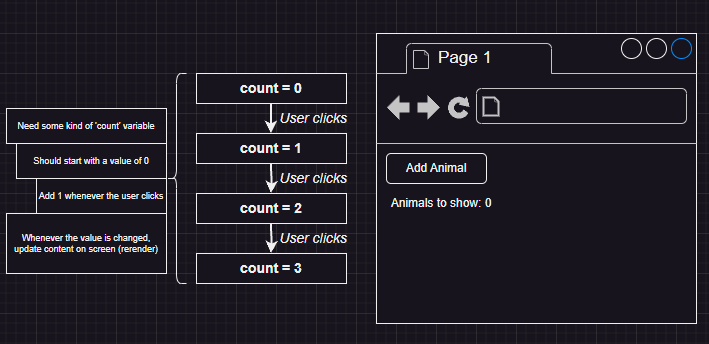
**Example Scenario**

**Suppose you have a list of items and you want to update one item:**

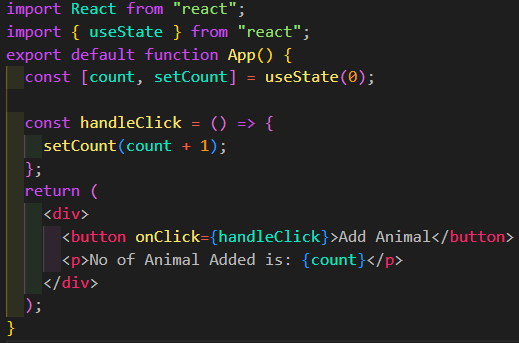
**DOM Approach: Directly update the item in the DOM, causing a potential reflow and repaint of the entire list.**

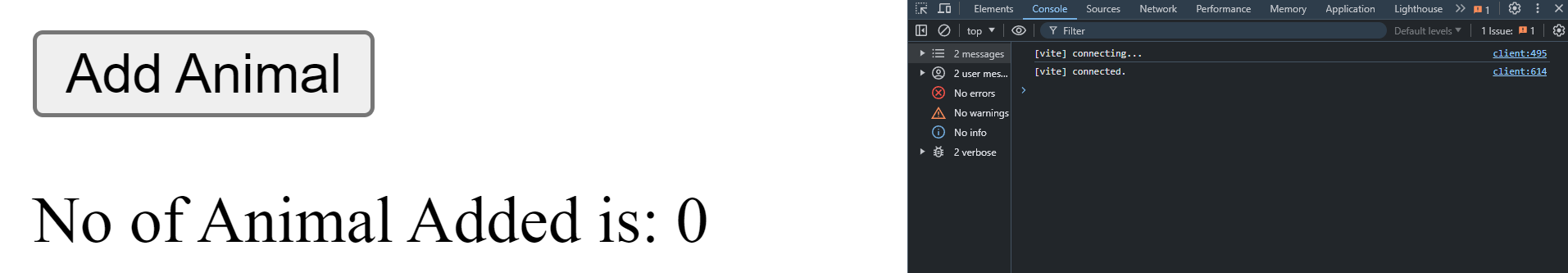
**VDOM Approach: Update the item in the Virtual DOM. React calculates the minimal changes required and updates only the specific item in the actual DOM, avoiding unnecessary reflows and repaints.**

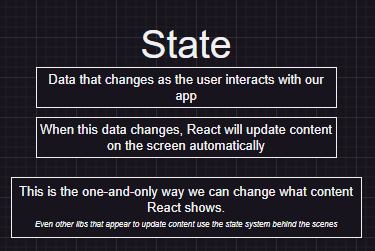
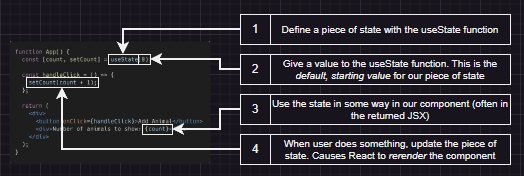
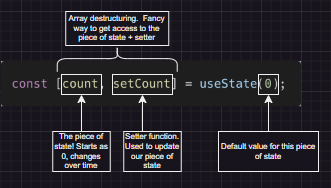
**In summary, the Virtual DOM provides a more efficient way to handle updates in web applications by minimizing the performance cost associated with direct DOM manipulations.**

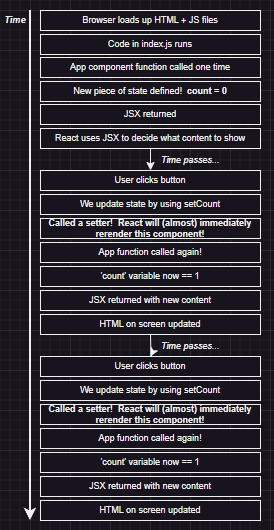
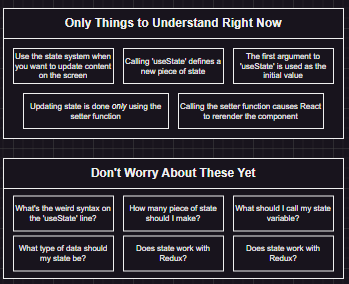
****

**Using State to update and re-render:**

****

**About State:**

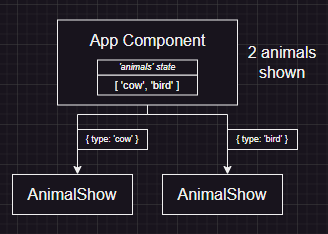
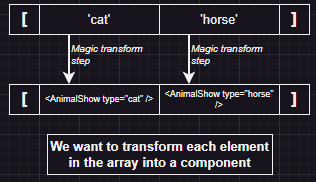
** **

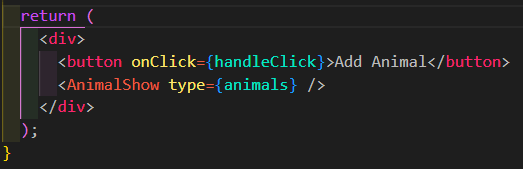
** **

**Whenever we call a state to change or update, React call the whole component that state is kept in. useState() return an array with 2st value is the default value and 2nd is the function, that’s why we de-structure it in [value, setValue]**

**Explain re-rendering in react in Details**

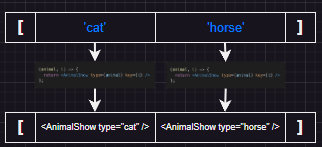
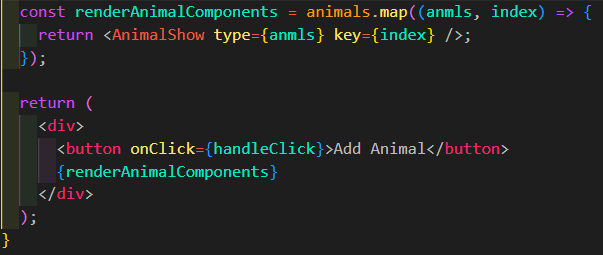
**Project discussion continue:**

** **

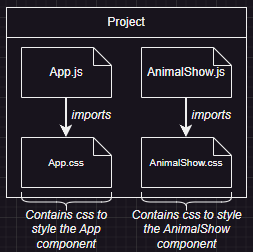
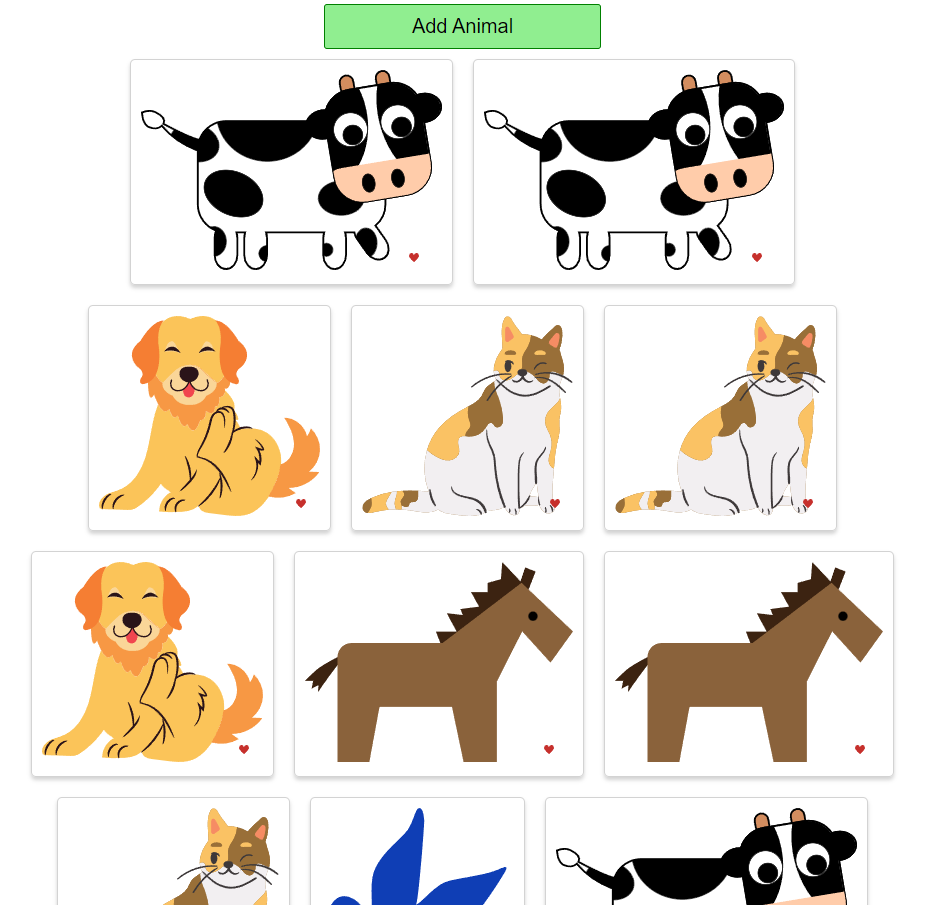
****

**Our default way to add a component to App.jsx but what when we have multiple versions of the same components.**

**How map is working:**

** **

**Creating 2 diffe. CSS diff. component makes code readable.**

** **

**How app looks when completed.**

**We used CSS to make the animal relative and heart absolute, why we did that?**

**Key Differences**

* **Document Flow**:
  + relative: Element remains in the document flow and affects the layout of other elements.
  + absolute: Element is removed from the document flow and does not affect the layout of other elements. So the heart will be attached to individual animal container and its growing will not affect the position on nearly placed other animal containers.
* **Positioning Reference**:
  + relative: Moves the element relative to its original position.
  + absolute: Moves the element relative to its nearest positioned ancestor (in this case animal-show it’s the ancestor of heart CSS).

**Practical Use**

* Use position: relative when you need to adjust an element's position while keeping it in the normal flow of the document.
* Use position: absolute when you need to precisely control the position of an element without affecting the layout of other elements.

Sasa

**Section 5: Using an API with react**