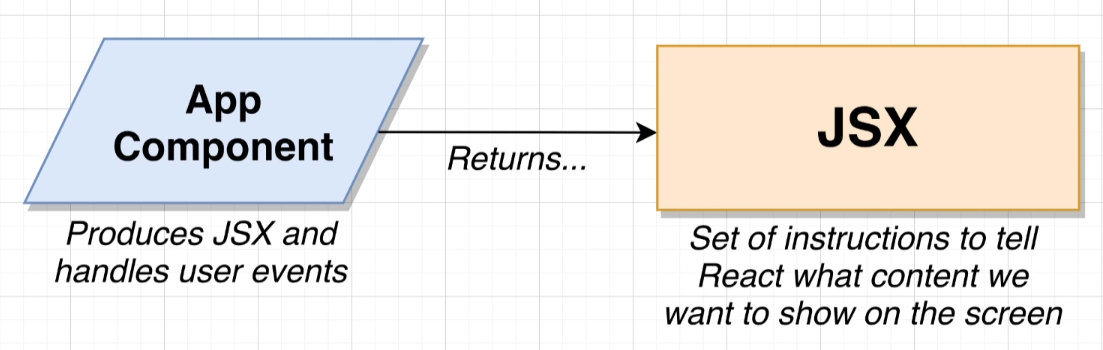
1. **What was the App function?**

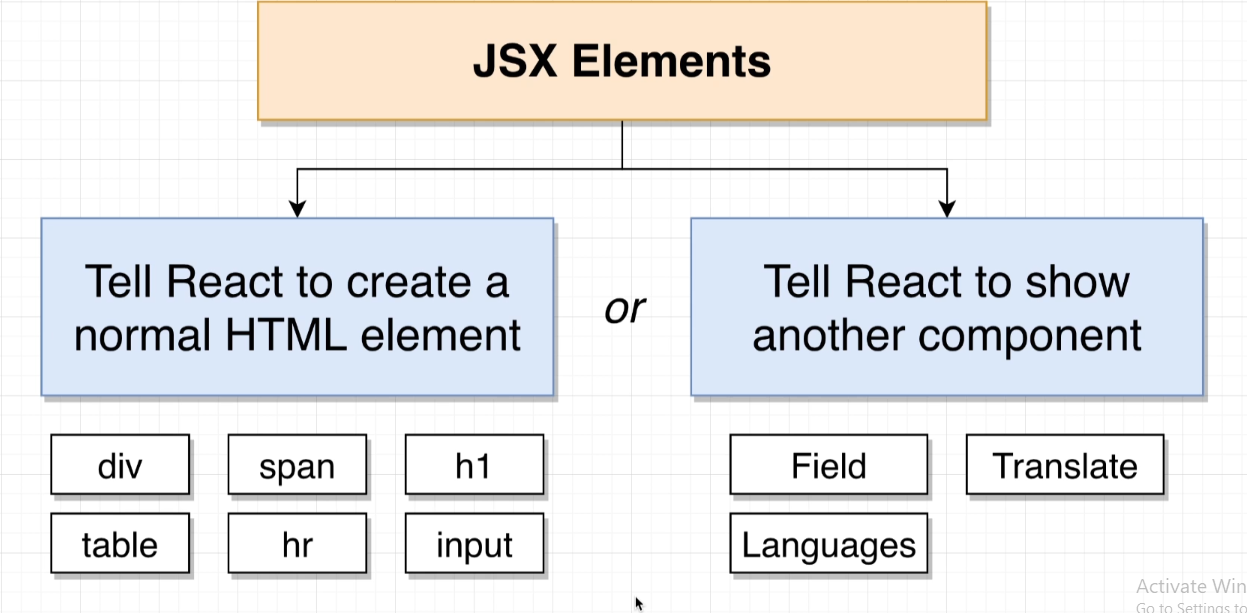
At the bottom, we’re returning what looks like HTML.

The App function is referred to as a React Component (plain JS function that returns some JSX).



JSX is similar to HTML in many forms. For each tag/element, we either have a closing element or a self-closing tag.

1. **What is JSX Elements all about?**



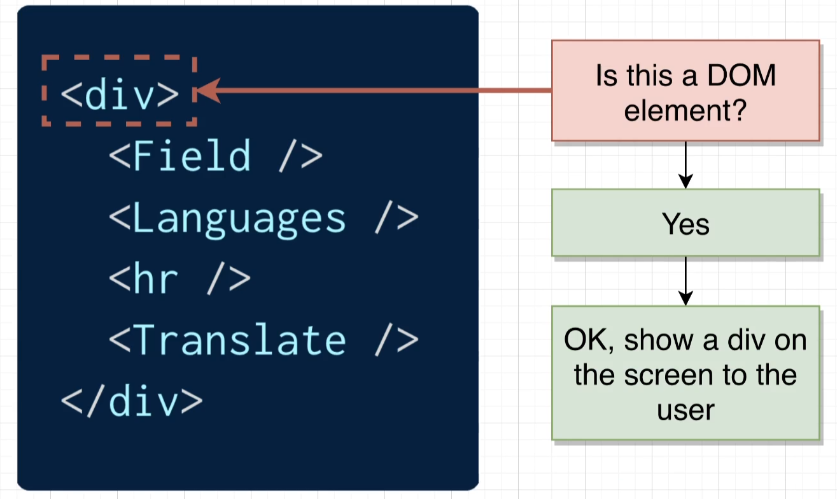
1. **What React does with the JSX Elements?**

Whenever we return JSX from a component, React starts to iterate over every element inside the block of JSX.

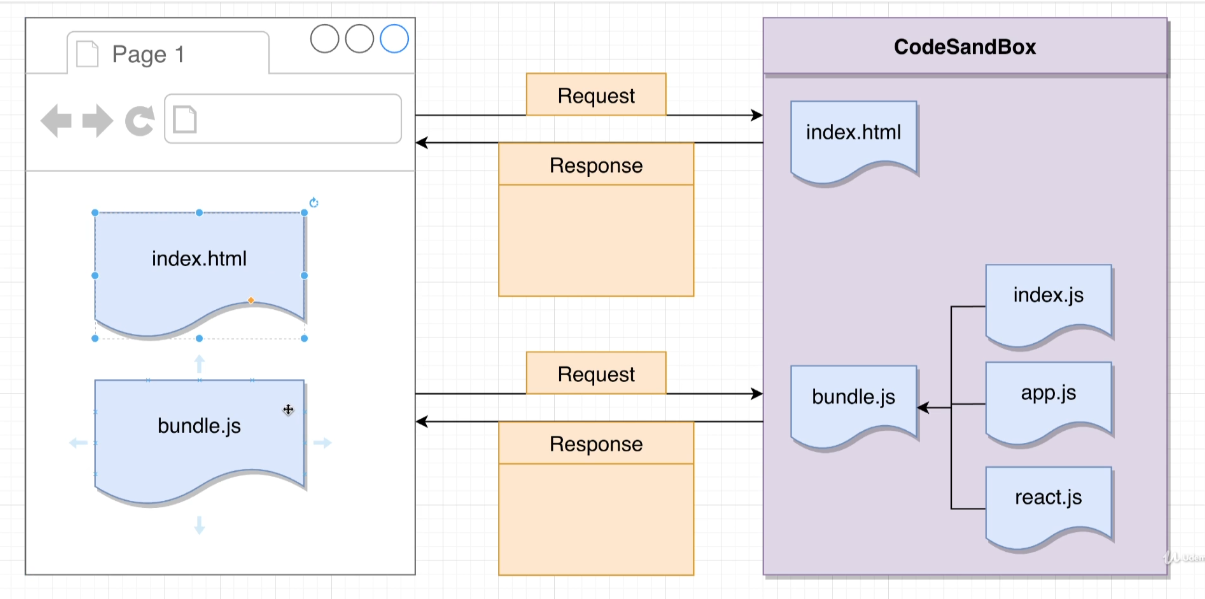
If **element is DOM element**, it shows it to the user.

If **not a DOM element**, React calls the component function and inspect all the JSX that comes back.

This is done until we’ve iterated over all the elements in the App function.



1. **How did content get displayed on screen?**

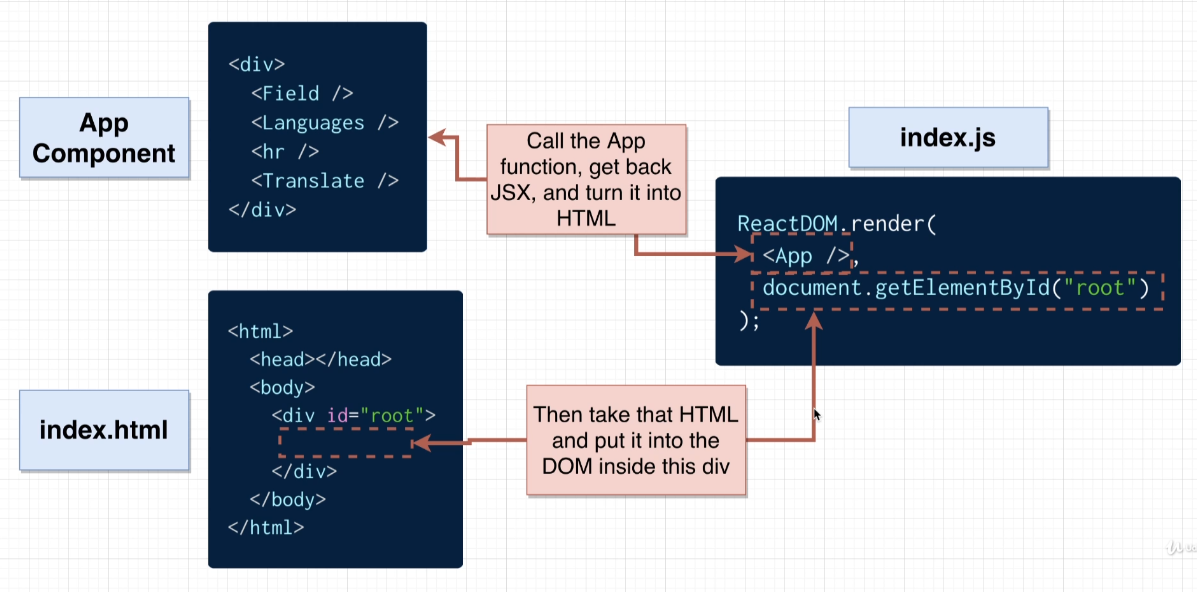


bundle.js – Special JS file combines all different JS files present in the project.

Mini browser window sends a request to code sandbox -> we get back index.html file -> this file has some script tags in it that causes the mini-browser window to make follow up request and gets a bunch of JS files that are then executed.

index.js – file and code inside it boots up React application and gets everything running.

1. **ReactDOM.render(…)?**



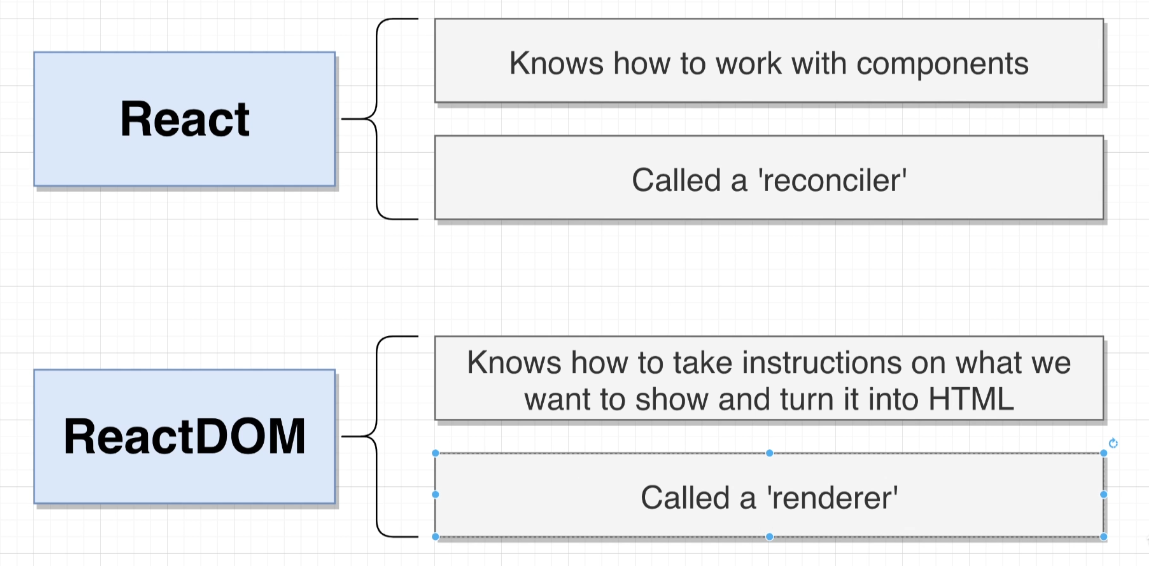
The second argument is a ‘reference’ to where we want to show our project inside index.html file.

1. **React v/s ReactDOM?**

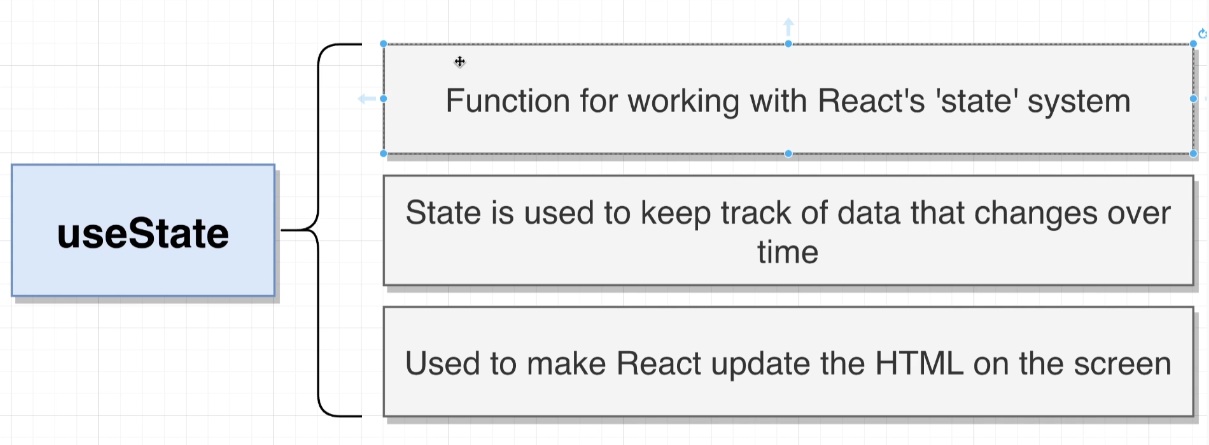
Whenever we work with React, we’re working with two separate libraries – React and React DOM.

React library itself has a bunch of code inside it that knows how to get a bunch of different components to work together, to call a component function, to get back JSX and iterate over all elements.

React DOM library is a special library that knows how to take a set of instructions of different elements we need to create and actually creates HTML out of it. (JSX -> HTML rendered in DOM)



1. **useState function?**



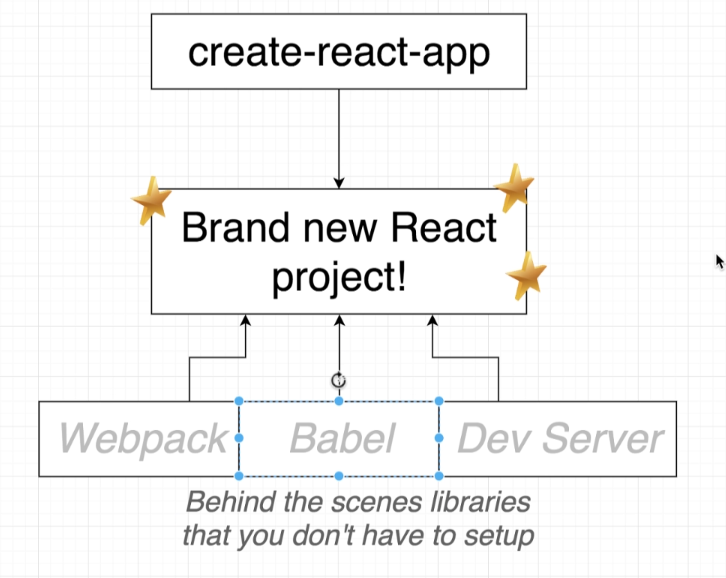
We’re trying to store some piece of data regarding the language user has chosen and text user has entered into the input field.

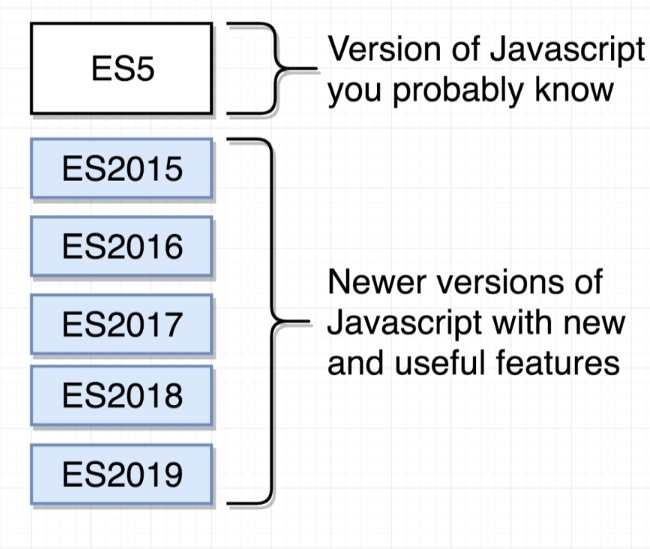
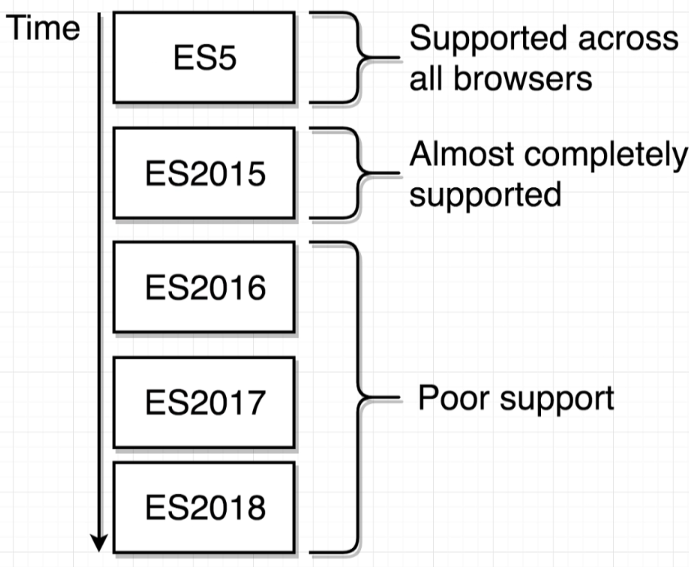
Both these values are going to change over time and thus, we use the ‘state’ system of React to keep track of that data.

It is used anytime we want to get React to somehow update the content on the screen.

1. **create react app?**

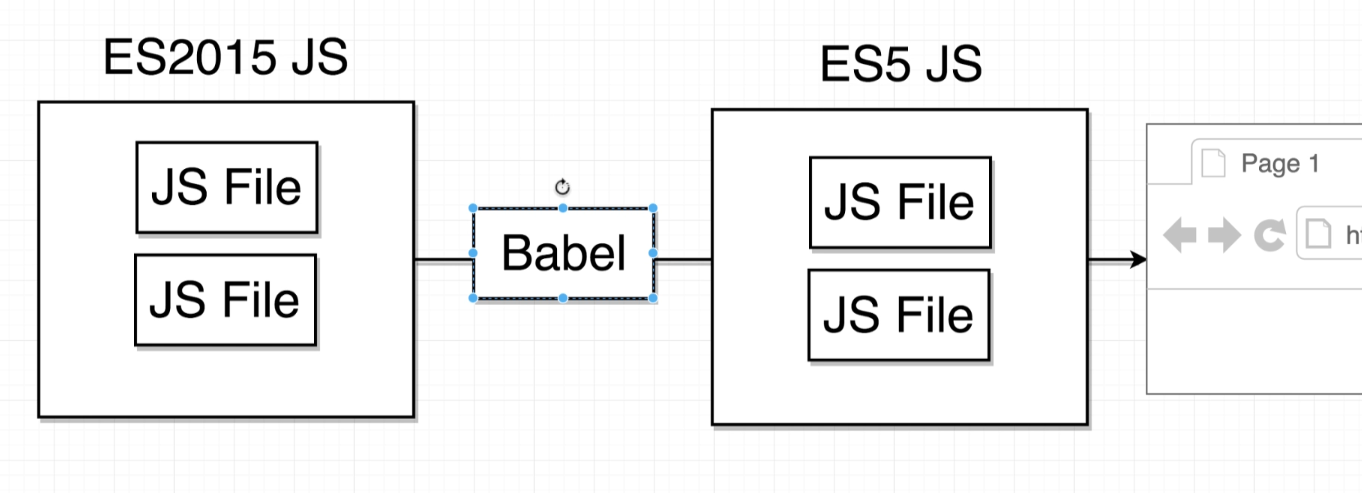
We use create-react-app to generate a brand-new React Project, but, inside it we have a ton of different dependencies that will help us properly write our application.



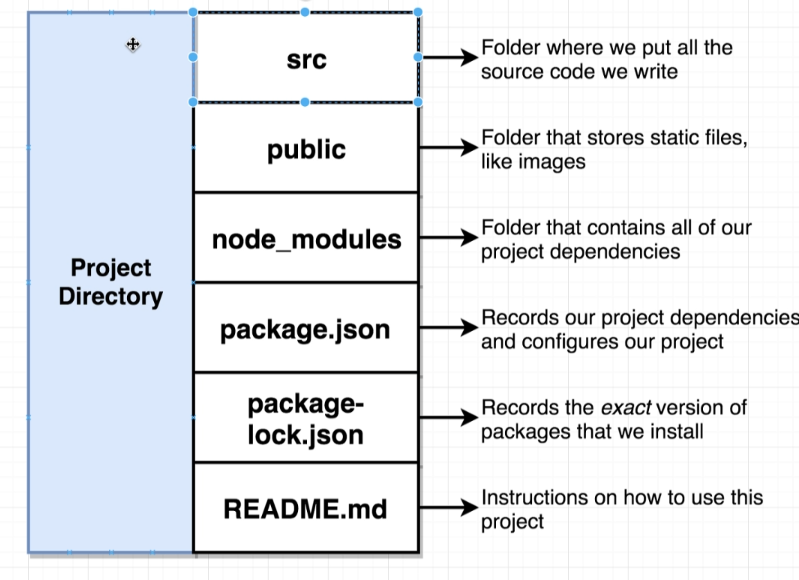
 

We want to use later versions to make our lives as developers easier, but on the other side, users will be unable to run the code because it is not supported by the browser.

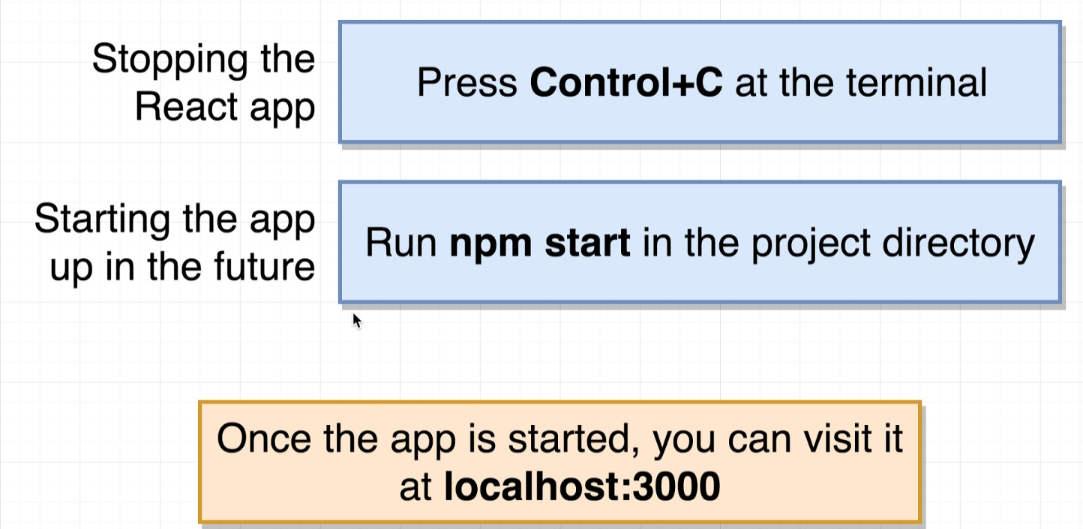
In order to get around with this, we make use of a command-line tool called ‘**Babel**’ – takes in any version of JS and spits out newer stable version that can run in specific user’s Browser.



1. **exploring the project directory –**



**.gitignore** – reference to the Git version control system, and essentially lists out different files and folders that Git should ignore while tracking our project i.e. while maintaining a record of all the different changes we’re making to our codebase.

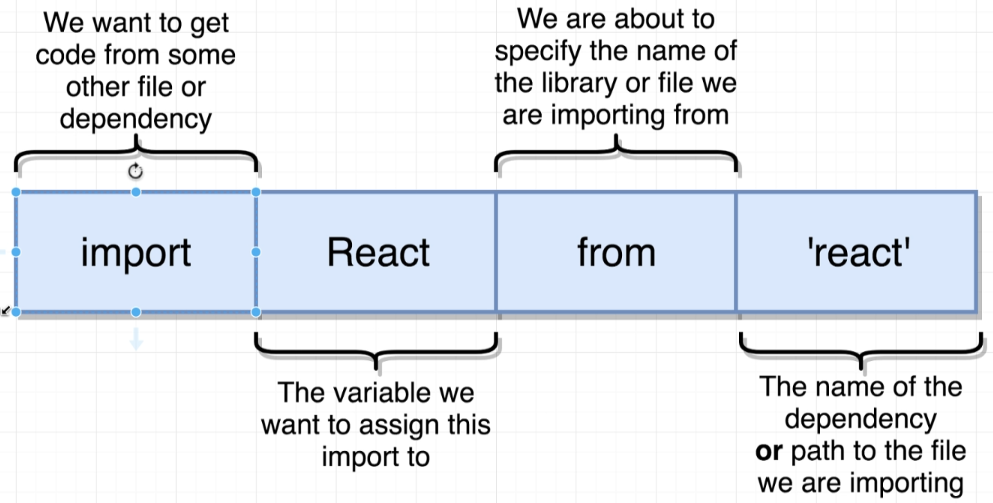


1. **get some content to appear on screen –**

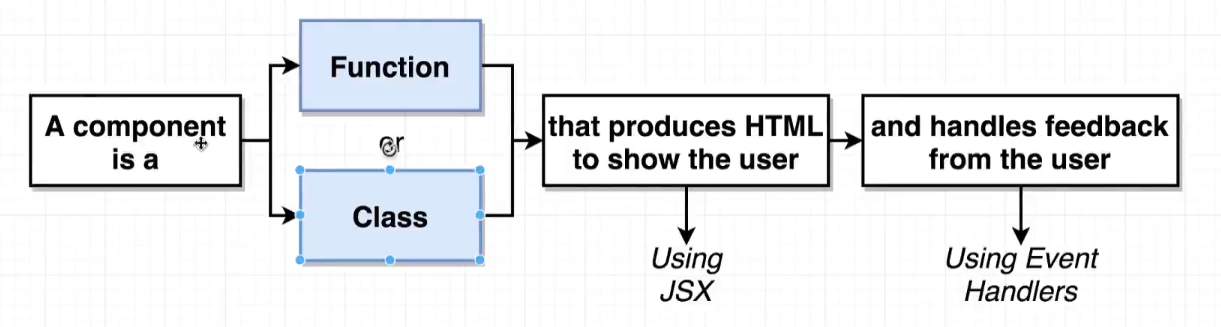
* Import the React and ReactDOM libraries

import React from ‘react’;

import ReactDOM from ‘react-dom’;



* Create a react component



const App = function() {

return <div> Hello There ! </div>;

};

* Take the react component and show it on the screen

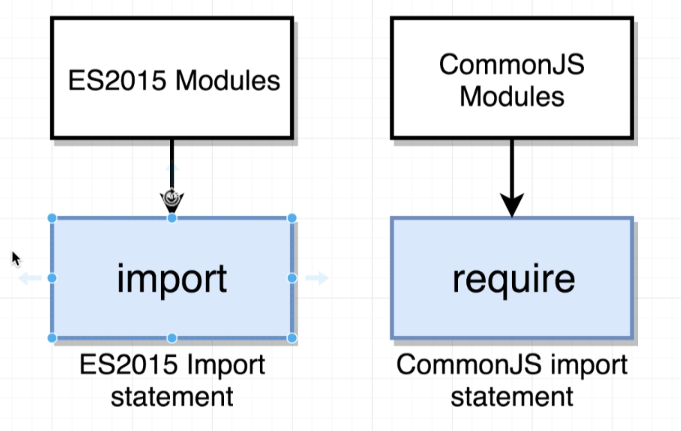
ReactDOM.render(

<App />,

document.querySelector(‘#root’);

);

1. **import vs require -**



Import – reference to a module system called ES2015 Modules. This module system is essentially a set of rules that describes how code can be shared between different JavaScript files.

Common JS – a different module system.

Essentially they are two different module system (rules) that do the same thing.