

1. INTRO TO REACT


Agenda:

- ① 'Hello World' → HTML, JS
- ② Install/Use React using 'CDN' & create 'Hello World'.
- ③ Create an element `<div>`
- ④ Create nested element `<div> <p> ... </p> </div>`
- ⑤ Use root.render
- ⑥ Functional Component
- ⑦ JSX → JavaScript + XML
- ⑧ How to read JSX (Babel)
- ⑨ Advantages of JSX.

Ways to get 'REACT' in your code

- ① CDN Links (Now)
- ② ~~Package managers (Build tools) (Later)~~

React ~~can~~

 `<script crossorigin src="https://unpkg.com/react@18/um`

`d@18/umd/react-dom.development.js"></script>`

React DOM CDN

✓ React : Provide framework for creating Components, manage them & it defines the structure & behaviour of components.

✓ React-DOM : Handles rendering of React Components to the DOM, making it visible to the browser.

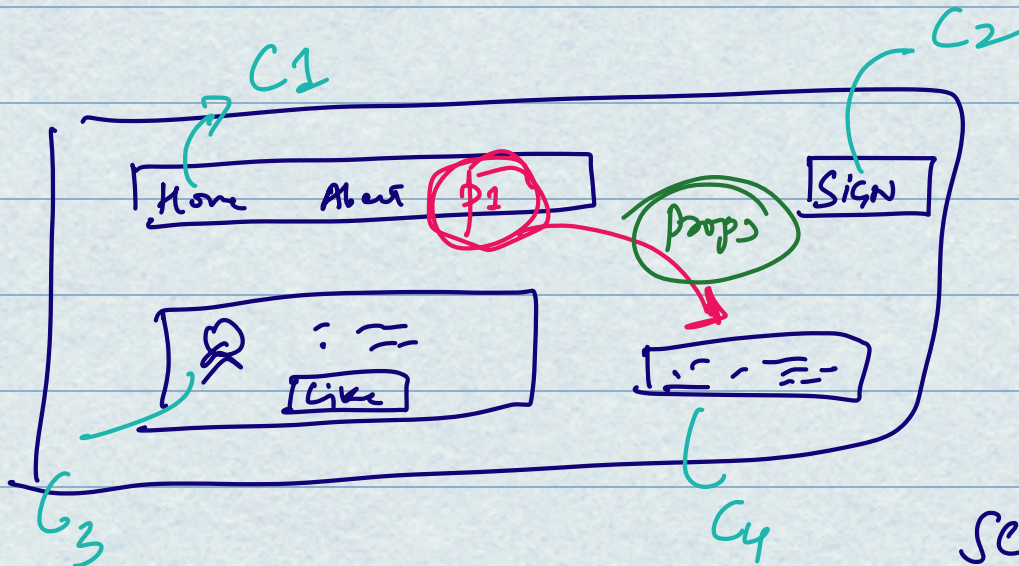
Manages updates to DOM when React makes changes.

Props:

React.createElement('h1', null, 'Hello from React');



props \Rightarrow Use props to pass any attribute from one Component to other.



Second Argument
used for properties
(props)

const heading = React.createElement('h1', {
 style: {color: 'blue'}
}, 'Hello from React');

• 'style prop'

JSX (Why JSX)??

→ With current (React. create ~), issues →

① Verbosity & Readability

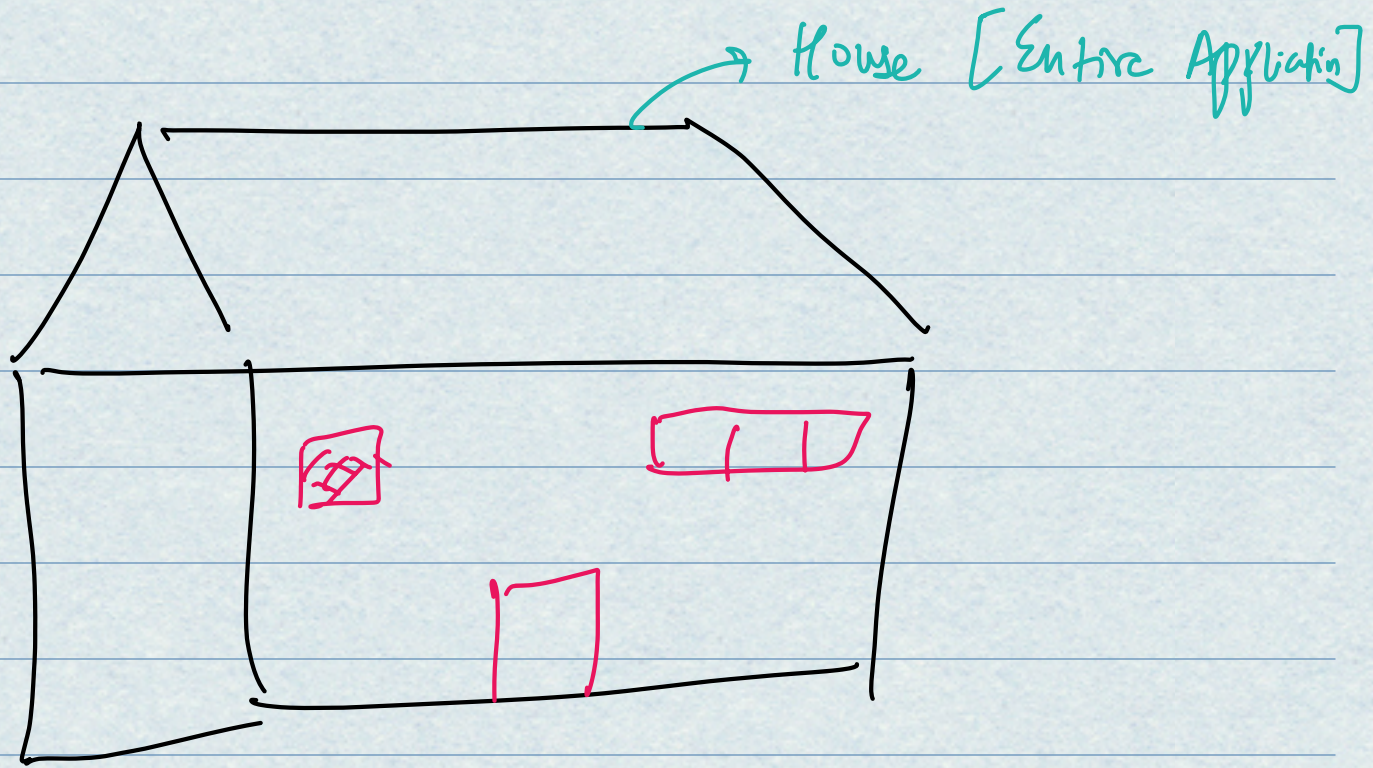
② Maintenance & Scalability.

③ Integration challenges

JSX → Practical development

React.createElement, { → To know how React
ReactDOM.render works under the hood.

JSX & Components



Component \Rightarrow independent piece of UI.

JS function \Rightarrow elements to show on screen.

① Class Components

Extends `React.Component`,
Can have complex logic,
lifecycle methods &
state management.

② Function Components

Simple JS functions
that return JSX.

JSX → extension of JS, that looks similar to XML/HTML.

→ function App() {
 return <h1>Hello from React JSX</h1> ← Returns JSX from Component.
}
ReactDOM.render(<App />, rootEl) // App Component.
↳ (React interacts with ReactDOM)

Annotations:
- JSX (green arrow pointing to `<h1>Hello from React JSX</h1>`)
- JSX (red arrow pointing to `<App />`)
- Instantiation (blue arrow pointing to `<App />`)

App → Functional Component in React.
JS function returns JSX (React element)

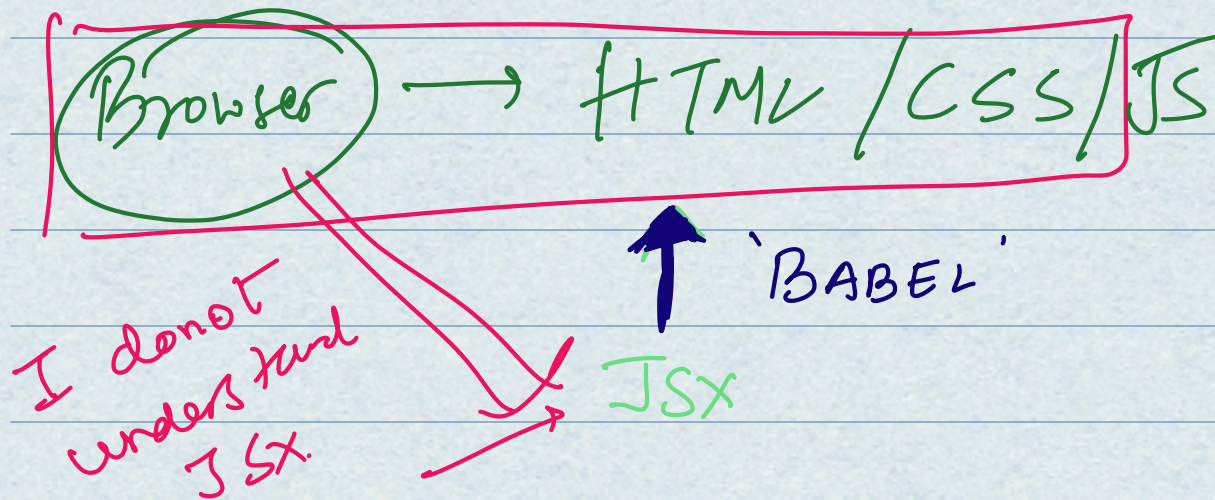
JSX → Allows to write HTML like syntax within Javascript.

`<App />` ⇒ JSX tag.

↳ Translates to React element representing the App Component.

↳ Similar to creating an instance of a class or

Calling the function.



Babel \Rightarrow JS Compiler that transforms syntax.

↳
TRANSPILER \leftarrow Translates
Compiler

```
function App() {  
  return <h1> Hello </h1>  
}  
  
React.createElement(  
  'h1', null, 'Hello'  
)
```

This transformation makes JSX syntax
usable in browsers that only understand
JavaScript.

① `<script type = 'text/babel' >`

↳ Allowed to write JS & JSX directly in our HTML files.

② Need Babel to Convert JSX into React calls.

ADVANTAGES of JSX :

- Familiar syntax

↳ HTML like syntax

↳ within JS files, allows to write HTML.

- Readability

↳ easier to develop components.

- Component Structure

↳ Visually evident what we are writing in code.

- much like HTML like layout

- Efficiency in development

React

ReactDOM

Component

JSX

Babel

props ??

Library & Framework

↓
React → View layer library
→ Choice is with devs.

↓
Angular.

- NO flexibility
- All the things included

API / Data
View / UI
Events

Framework

↳ All Capabilities are present

NextJS → State, props, Build tools.