# EPISODE 3 : LAYING THE FOUNDATION

## PART-1:

## ADDING SCRIPTS:

- To start the app we used npx parcel index.html , writing it again and again will cause difficulty
- So lets create scripsts for dev built and production built
- In package.json file in scripts add the following

```
5   "scripts": {
6     "test": "jest",
7     "start": "parcel index.html", //added this line
8     "build": "parcel build index.html"   //added this line      You, 1 second ago • Uncommitted changes
9   },
```

- Now u can use ***npm run start*** or ***npm start*** (this shortcut only works with start because it is a predefined react keyword it not work if we execute npm build)instead of using npx parcel index.html.(fordev built)
- And npm run build for production built.

```
shanmukha@shanmukha-Latitude-3420:~/Music/the_react$ npm run start

> the_react@1.0.0 start /home/shanmukha/Music/the_react
> parcel index.html

Server running at http://localhost:1234
✨ Built in 5ms
shanmukha@shanmukha-Latitude-3420:~/Music/the_react$ npm start

> the_react@1.0.0 start /home/shanmukha/Music/the_react
> parcel index.html

Server running at http://localhost:1234
✨ Built in 5ms
```

```
shanmukha@shanmukha-Latitude-3420:~/Music/the_react$ npm run build

> the_react@1.0.0 build /home/shanmukha/Music/the_react
> parcel build index.html

✨ Built in 906ms

dist/index.html                 421 B    306ms
dist/index.30faf25f.css          81 B     90ms
dist/index.4202ffd5.js     141.85 KB    333ms
dist/index.a010ae67.js     141.06 KB    333ms
shanmukha@shanmukha-Latitude-3420:~/Music/the_react$ npm build
npm WARN build `npm build` called with no arguments. Did you mean to `npm run-script build`?
shanmukha@shanmukha-Latitude-3420:~/Music/the_react$
```

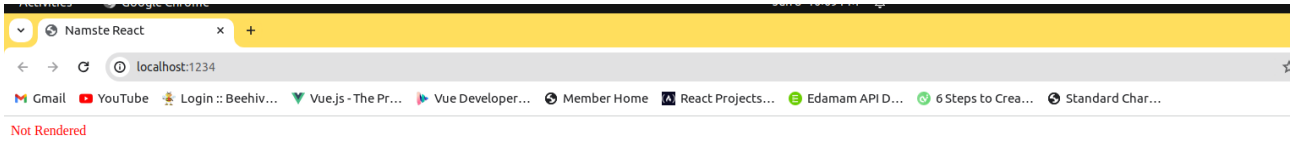
# PART 02:

- Normally we used React.createElement to create react element, this React.createElement is a object
- when it rendered ,the reactDOM converts it from object to html element.
- Lets remove everything from the app.js and check the output.

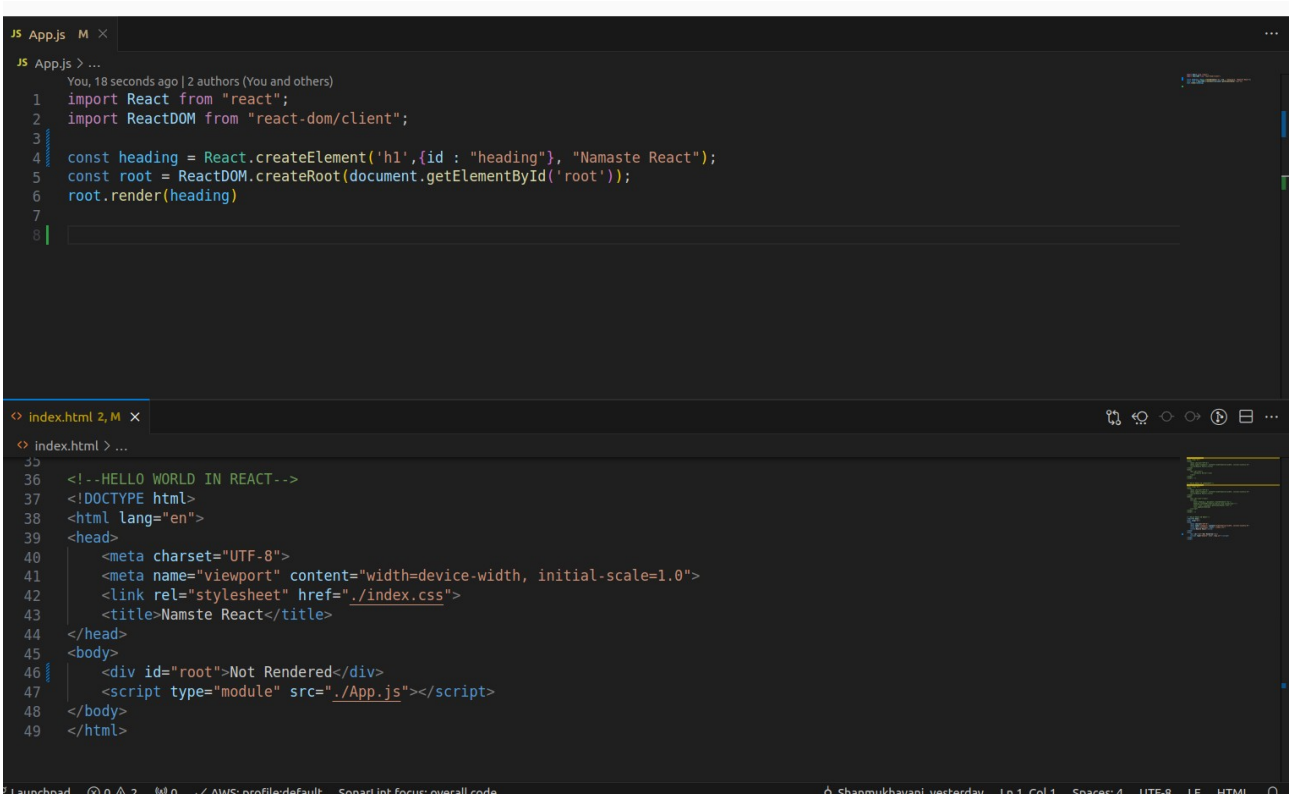
```
index.html > html > body > div#root
<!--HELLO WORLD IN REACT-->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="./index.css">
    <title>Namste React</title>
</head>
<body>
    <div id="root">Not Rendered</div>
    <script type="module" src="./App.js"></script>
</body>
</html>
```

```
App.js M
App.js
You, 2 minutes ago | 1 author (You)
import React from "react";      You, 5 hours ago • episode2/igniting_our_app
import ReactDOM from "react-dom/client";
```

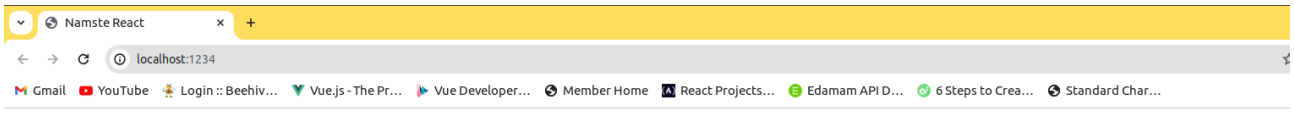
- The output will be the not rendered which is in index.html




- Now create any react element in app.js and try to render it inside the root of index.html
- You can observe react element replaces the html element.




# OUTPUT:




**Namaste React**

- In the output you can observe not rendered is replaced with Namaste React.

**PART 03:(17:06)**
**Introduction to JSX**

- Till now we have used react elements to create html elements like h1 etc.,
- The syntax is clumpsy so ***JSX*** is introduced.

**JSX :**

- JSX is not a html syntax it is html like syntax.
- We can say jsx is a combination of html and js but not as html.
- Browsers only understands js code but jsx is not a js code so before rendering to the browser the jsx is converted to js engine understandable code with the help of ***babel***

**BABEL:**

- **Babel** is a javascript transpiler it compiles jsx to javascript engine understandable code.
- Babel is a toolchain that is mainly used to convert ECMAScript 2015+ code into a backwards compatible version of JavaScript in current and older browsers or

environments. Here are the main things Babel can do for you:

•Transform syntax

•Basically REACT element is a JS object ->when it renders it converts to html and renders in browser.

*REACT ELEMENT(JS OBJ)->renders->HTML*

•*jsx->(babel)->react element->html*

•*When you write jsx in console you will see browser wont understand it.*

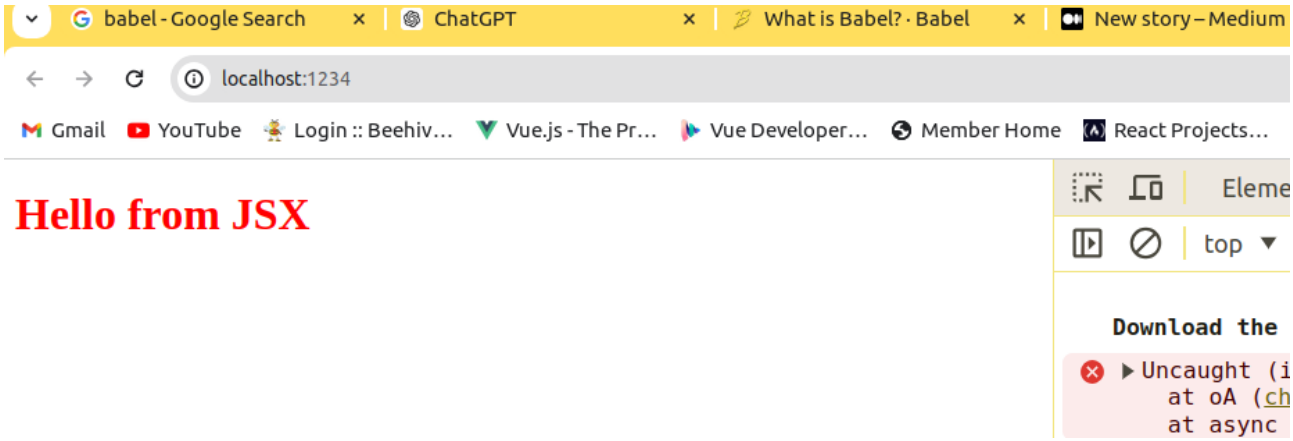
```
Download the React DevTools for a better development experien
Uncaught (in promise) Error: 400: Invalid Token
    at oA (chrome-extension://d…ckground.js:5:17886)
    at async chrome-extension://d…ckground.js:1:41135
> const heading = <h1>HELLO<h1>
Uncaught SyntaxError: Unexpected token '<'
>
```

•*Babel compiles jsx token by token*

**WRITING OUR CODE IN JSX**

```
App.js > ...
You, 5 seconds ago | 1 author (You)
import React from "react";
import ReactDOM from "react-dom/client";

// const heading = React.createElement('h1',{id : "heading"}, "Namaste React");
// const root = ReactDOM.createRoot(document.getElementById('root'));
// root.render(heading)

const JSXheading = <h1>Hello from JSX</h1>
const root = ReactDOM.createRoot(document.getElementById('root'))
root.render(JSXheading)        You, 1 second ago • Uncommitted changes
```

**output:**




•Here JSXheading is an object

•JSX different from html because the atrributes in JSX are different when compared to html.

•In html we give the attribute as class but in jsx it is className

•We need to wrapup jsx with rounded braces when we are writing multiline jsx code no need for single line code.

• Ex:single line jsx

Const jsxheading =<h1 id=”heading”>I’m h1 tag with jsx</h1>

• Ex:multiline jsx

Const jsxheading =( <h1 id=”heading”>

I’m h1 tag with jsx

•</h1>)

**EPISODE 3: PART4**

**COMPONENTS**

- In react everything is a component that means a button, card , nav bar
- everything is a component
- Two types of components
- Class based components(oldway)
- Functional components(new way)

React Functional component is just a normal javascript function which returns a jsx code or React element.

•Name the react components with capital letter.

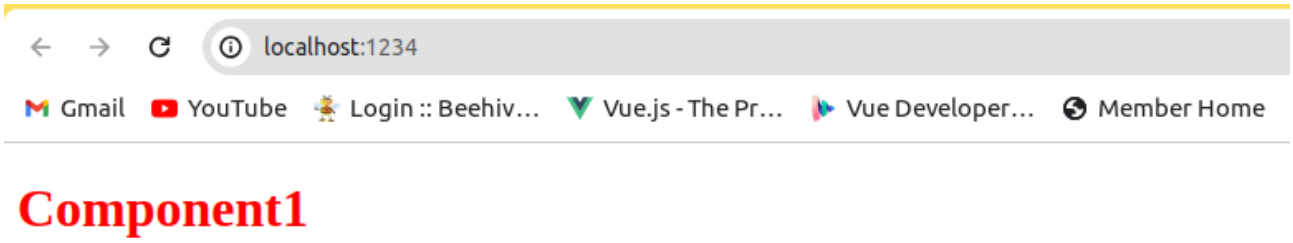
**Syntax of functional component :**

```
const func = () => {
    return true
}

const func2 = () =>{
    true
}

const func3 = () => true

const Component1 =()=>{
    return <h1>Component1</h1>
}

const Component2 =()=>{
    <h1 className="heading">Component2</h1>
}

const Component3 = () => <h1 className="heading">Component3</h1>

const Component4 = () =>(
    <h1 className="heading">Component4</h1>
)

const Component5 = () =>(
    <h1 className="heading">
        Component5
    </h1>      You, 1 second ago • Uncommitted changes
)
```

# RENDERING A REACT COMPONENT:

# OUTPUT :

```
JS App.js > ...
    You, 25 seconds ago | 2 authors (You and others)
1   import React from "react";
2   import ReactDOM from "react-dom/client";      You, 23 hours ago • episo
3   const Component1 =()=>{
4       return <h1>Component1</h1>
5   }
6
7   const root = ReactDOM.createRoot(document.getElementById('root'));
8   root.render(<Component1/>)
9
```




## Component Composition :

Renderning component within other component is called as Component Composition.

```
JS App.js > Component2
You, yesterday | 2 authors (You and one other)
import React from "react";
import ReactDOM from "react-dom/client";




const Component1 =()=>{
    return <Component2/>
}

const Component2 = () =>{
    return <h1>Component Composition</h1>      You, yesterday • Uncommitted changes
}

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(<Component1/>)
```

In the above example we are calling component2 in component1.

# OUTPUT:


**Component Composition**

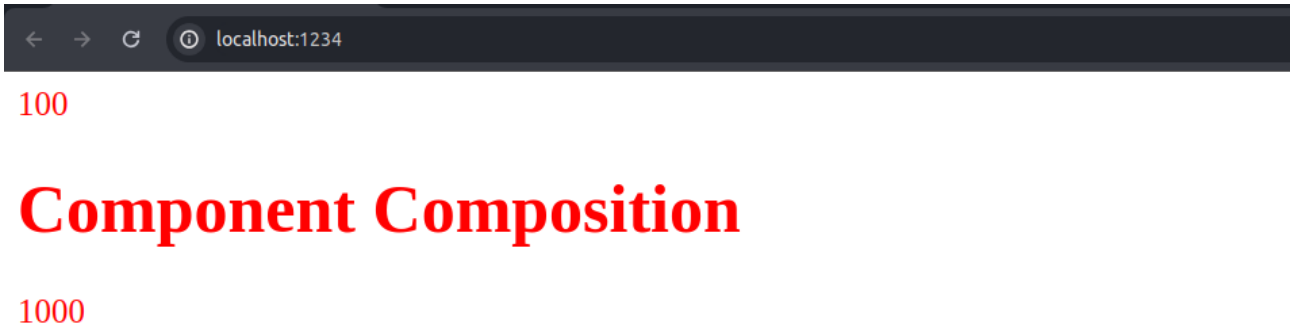
# PART 05:

## Using JavaScript code inside jsx:

We can write javascript code inside the jsx in curly braces{}

```
App.js > ...
You, 1 second ago | 2 authors (You and one other)
import React from "react";
import ReactDOM from "react-dom/client";

const number = 1000;
const Component2 = () =>{
    return <h1>Component Composition</h1>
}
const Component1 =() =>{

   return(
    <div className="container">
        {100}
        <Component2/>
        {number}
    </div>

   )

}
      You, 1 second ago • Uncommitted changes
const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(<Component1/>)
```

**OUTPUT**:




In jsx we can call component inside another component ,element inside other element,element inside component ,component inside element.

**Jsx sanitizing:**

Suppose if the api sends a malicious data , the data variable holds
that content and we are executing {data} in jsx , but before executing
the code inside curlybraces{} jsx sanitizes the code whether its good
or not.

```
import ReactDOM from "react-dom/client";

const data = api.fetch();
const Component2 = () =>{
    return (
        <div>
            {data}        You, 1 second ago • Uncommitted
            <h1>Component Composition</h1>
        </div>
    )

}
const Component1 =() =>{

```

**Ways of calling a component:**

- <Component/>
- <Component></Component>
- {Component()}->because Component is just a normal javascript function we can call component just like a normal javascript function.

All the three things ate same