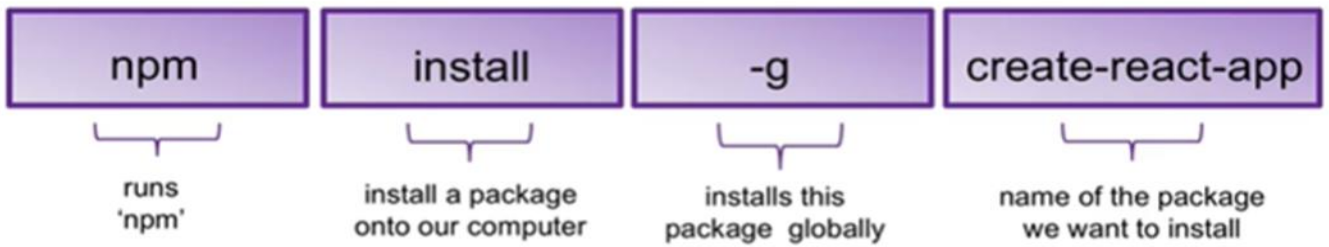
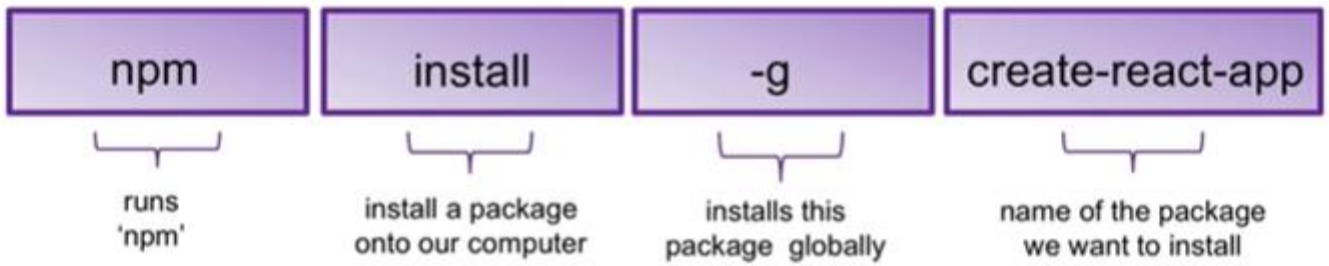
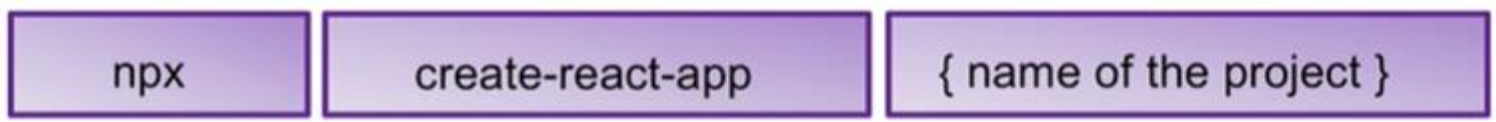
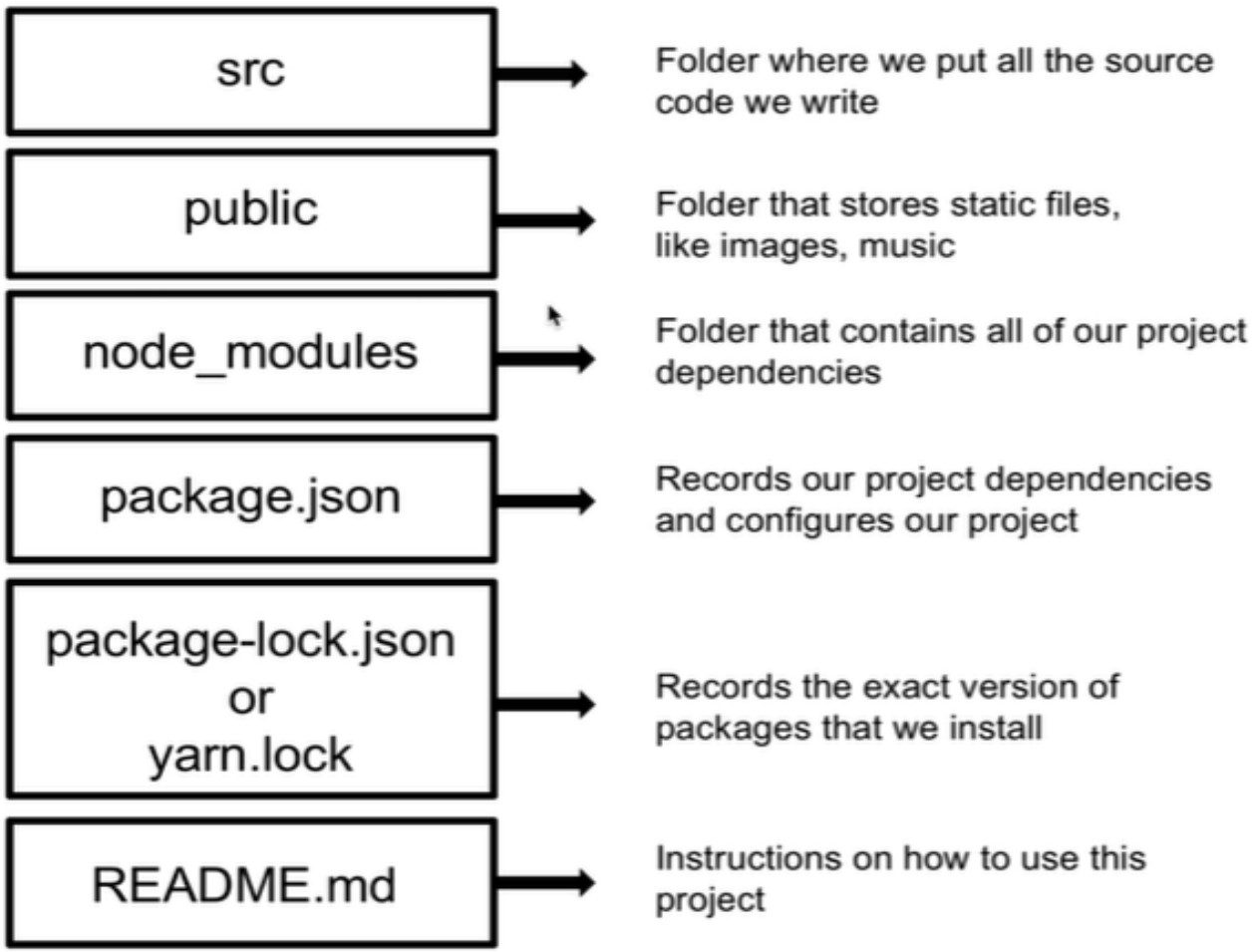

npm
install
-g
create-react-app
runs
'npm'
install a package
onto our computer
installs this
package globally
name of the package
we want to install


npm
install
-g
create-react-app
runs
'npm'
install a package
onto our computer
installs this
package globally
name of the package
we want to install


npx
create-react-app
{ name of the project }

Project Directory



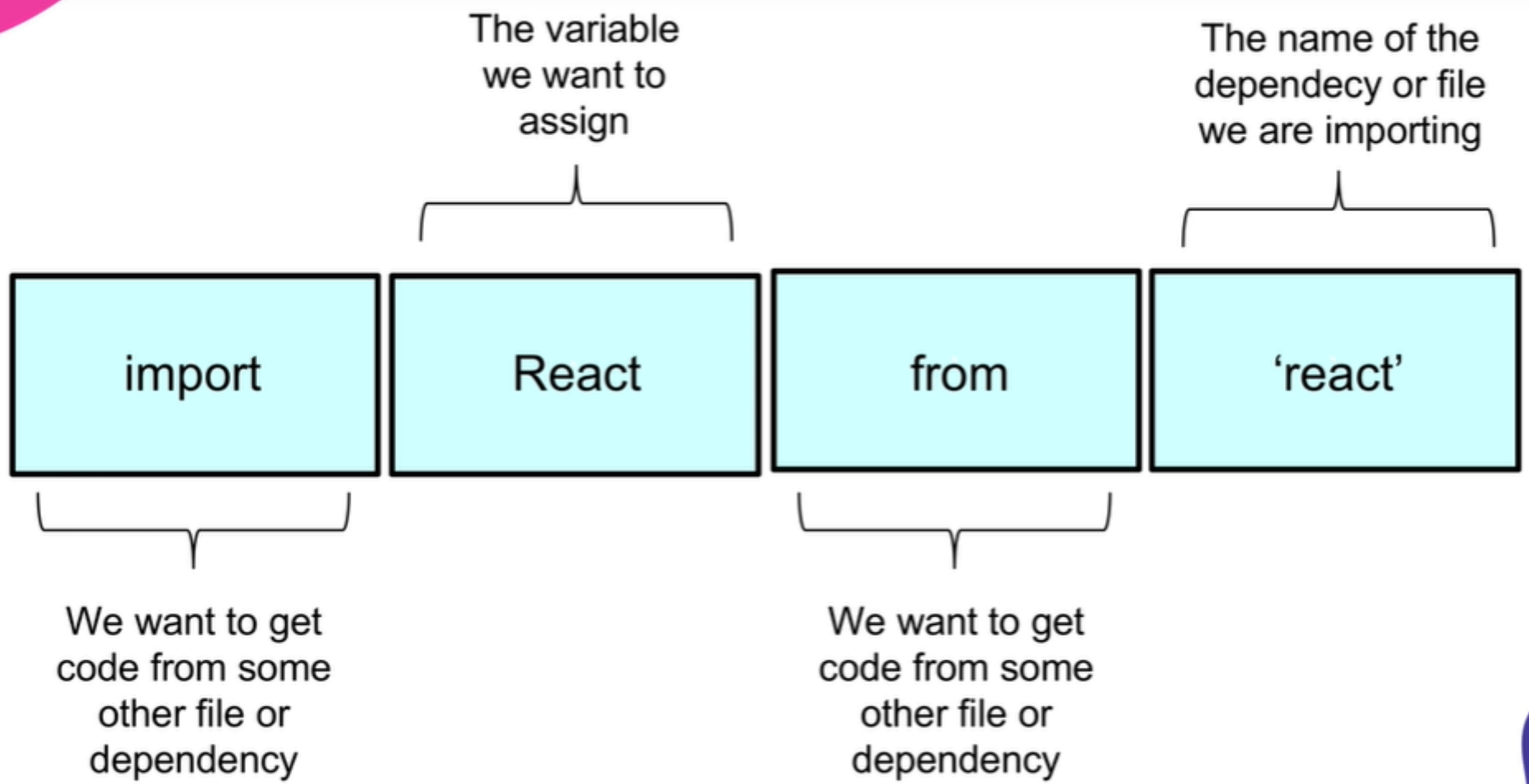
Stops the React App →

Press **Control+C** at the terminal

Starts the React App →

Run **npm start** in the **project directory**

Once the app started you can visit it at **localhost:3000**


The variable we want to assign
The name of the dependecy or file we are importing
import
React
from
'react'
We want to get code from some other file or dependency
We want to get code from some other file or dependency

How to style JSX?

Adding custom styling to an element

Adding a class to an element

JSX can reference JS variables

How to style JSX?

HTML

```
<div style=“background-color: red;”></div>
```

JSX

```
<div style={{ backgroundColor: ‘red’ }}></div>
```

Press Esc to exit full screen

Creating Reusable Component

Identify the Jsx that appears to be dublicated

Think of a descriptive name for reusable component

Create a new file for reusable component

Create a new component in the new file

Make the new component configurable by using ‘props’ system

Take any rep code is JSX

Then

Create Module (file)

Import React from ‘react’

- Cut and past the repeated code or the JSX to insert into as a component.
- It will be placed in the return of a function expression or arrow function.

Then

Export the function (export default <*name of function*>)

Then

Import to the main page

The import component is treated

As a JSX tag

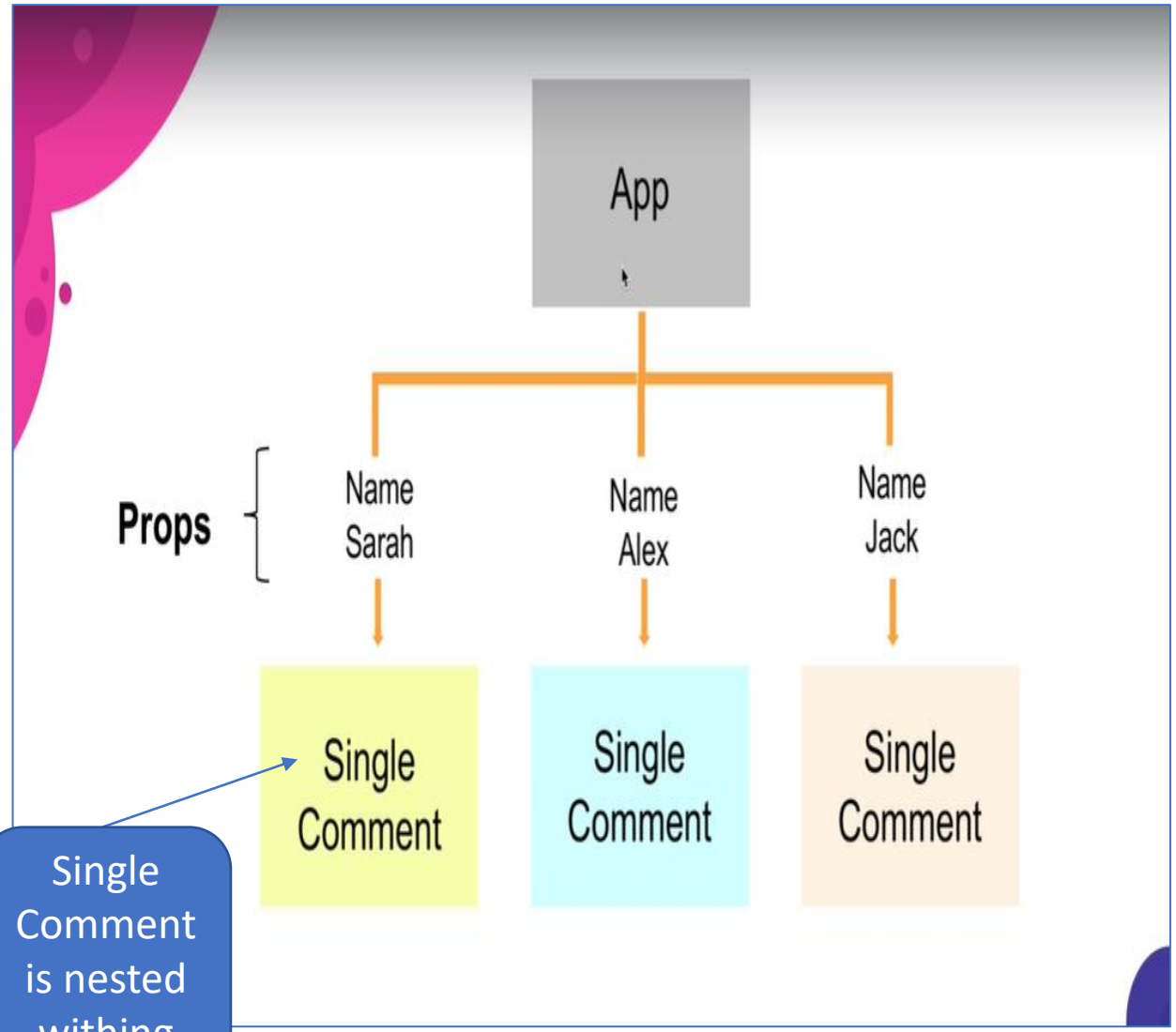
<SingleComent/>

PROPS

Component Nesting — A component can be shown inside of another component

Component Reusability — We want to make Components reused throughout application

Component Configuration — We should be configure a component when it is created



Single Comment is nested withing the app

Child

```jsx
import React from "react";

const UserCard = props => {
  return (
    <div class="ui card">
      <a href="/" className="images">
        <img src={props.picture} style={{ width: "100%" }} alt="profile" />
      </a>
      <div class="content">
        <a href="/" className="header">
          {props.name}
        </a>
        <div class="description">{props.description}</div>
        <div class="meta">
          <span class="date"> {props.date} </span>
        </div>
        <div class="extra content">
          <div className="text">{props.text}</div>
        </div>
      </div>
    </div>
  );
};

export default UserCard;
```

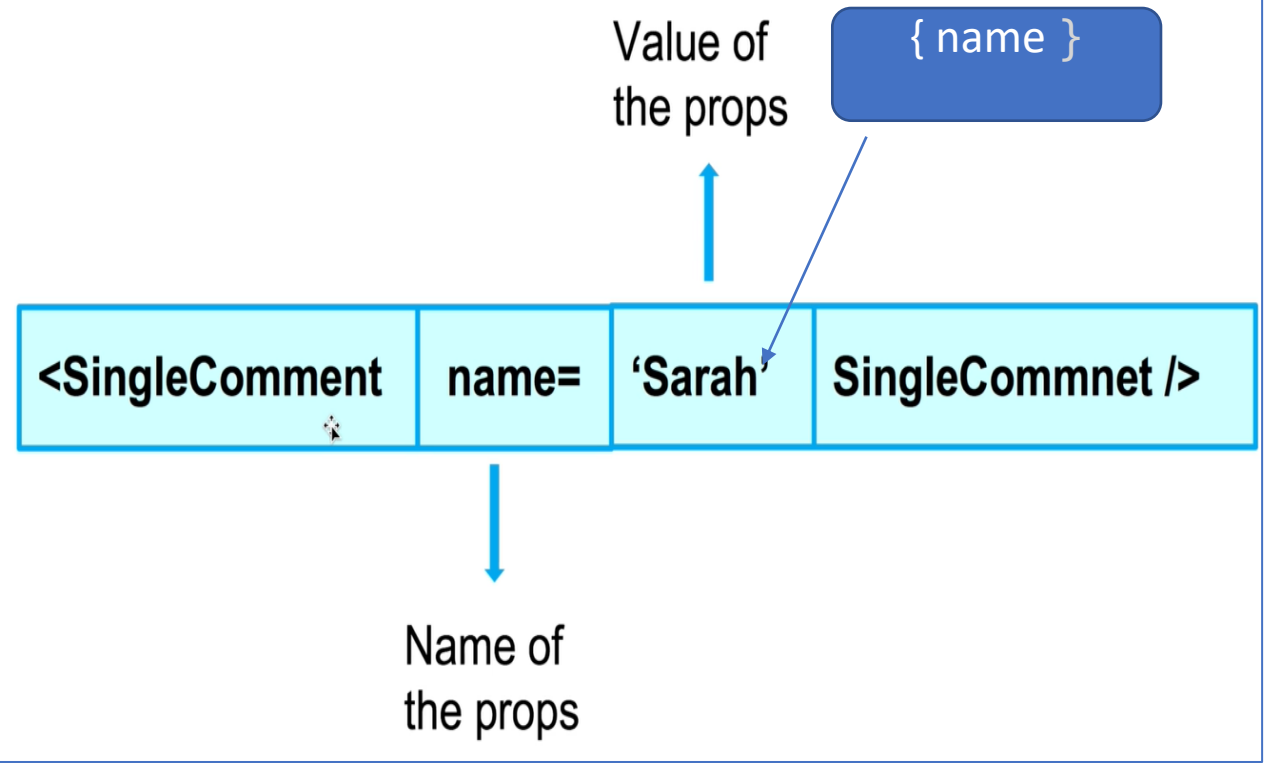
index.js × UserCard.js

```jsx
import React from "react";
import ReactDOM from "react-dom";

import SingleComment from "./SingleComment";
import UserCard from "./UserCard";

import profile1 from "./image/pic-1.jpg";
import profile2 from "./image/pic-2.jpg";
import profile3 from "./image/pic-3.jpg";

const App = () => {
  return (
    <div className="ui comments">
      <UserCard
        name="San Diego"
        date="9/3/2018"
        text="Vacation"
        description="Family Vacation"
        picture={profile1}
      />
      <UserCard
        name="Getty Museum"
        date="Sunday, December 26, 2021,"
        text="Vacation"
        description="Dad and Son"
        picture={profile2}
      />
      <UserCard
        name="San Diego"
        date="9/3/2018"
        text="Vacation"
        description="Family Vacation"
        picture={profile3}
      />
    </div>
  );
};

ReactDOM.render(<App />, document.querySelector("#root"));
```

Props is a system for passing data from a parent component to a child component

Props purpose is to customize or configure a child component

Props

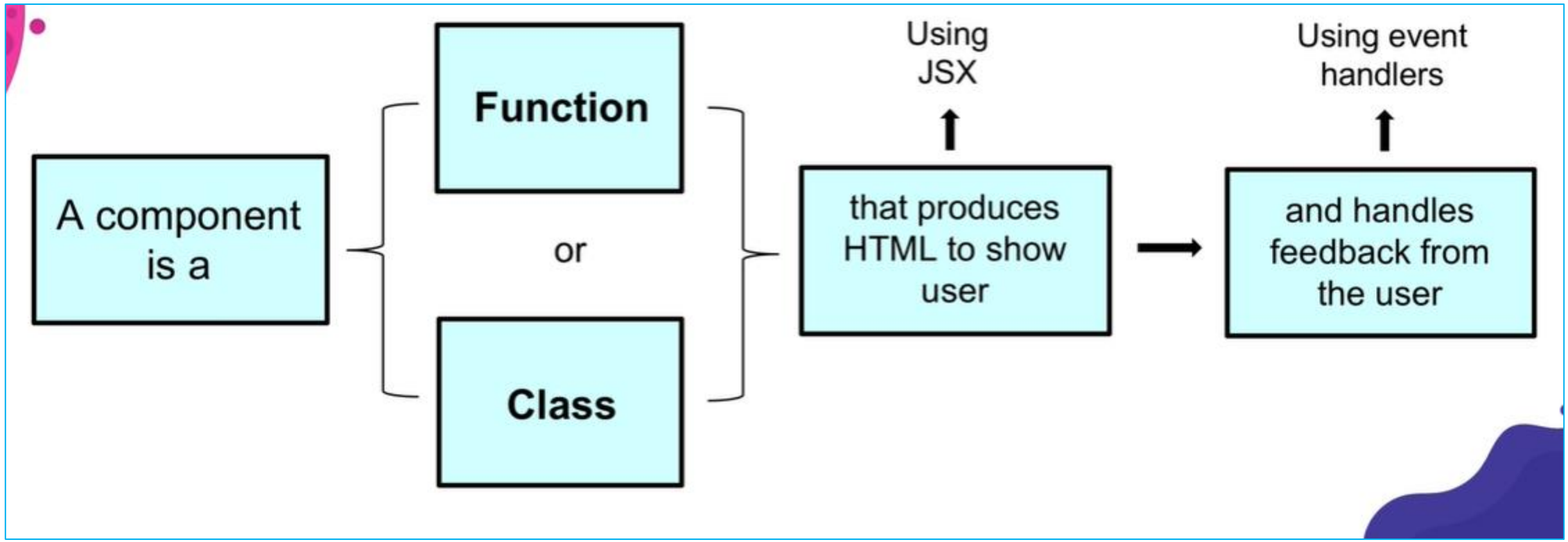

Value of
the props
{ name }
<SingleComment
name=
'Sarah'
SingleCommnet />
Name of
the props

JS index.js JS SingleComment.js

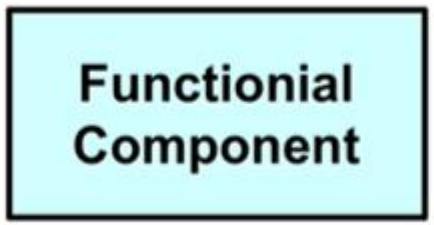
```
1  import React from "react";
2
3  const SingleComment = props => {
4    return (
5      <div className="u  card">
6        <div className="co ent">
7          <div className="head "> {props.children} </div>
8          <div className="descrip n"> A single Comment </div>
9        </div>
10       <div className="content">
11         <div>Notes: SingleComment encapsol s the usercard JSX Compoent</div>
12       </div>
13     </div>
14   );
15 };
16
17 export default SingleComment;
```

Children object

```
<SingleComment>
  <UserCard
    name="San Diego"
    date="9/3/2018"
    text="Vacation"
    description="Family Vacation"
    picture={profile1}
  />
</SingleComment>
```

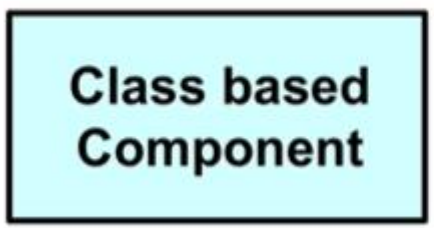

A component is a
Function
or
Class
that produces HTML to show user
Using JSX
and handles feedback from the user
Using event handlers

Semantic-ui.com



We can use it for simple content

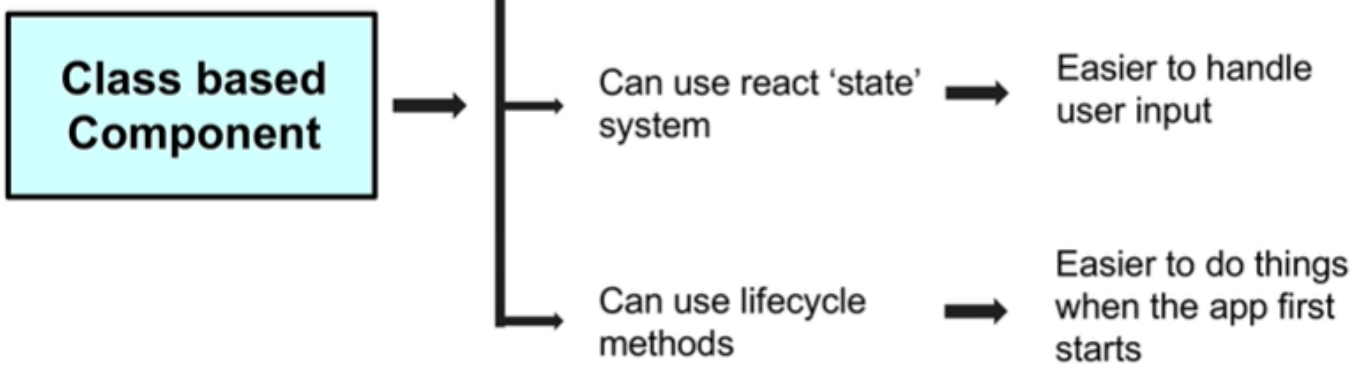
Not a lot of logic
Simple content
Stick with functional component

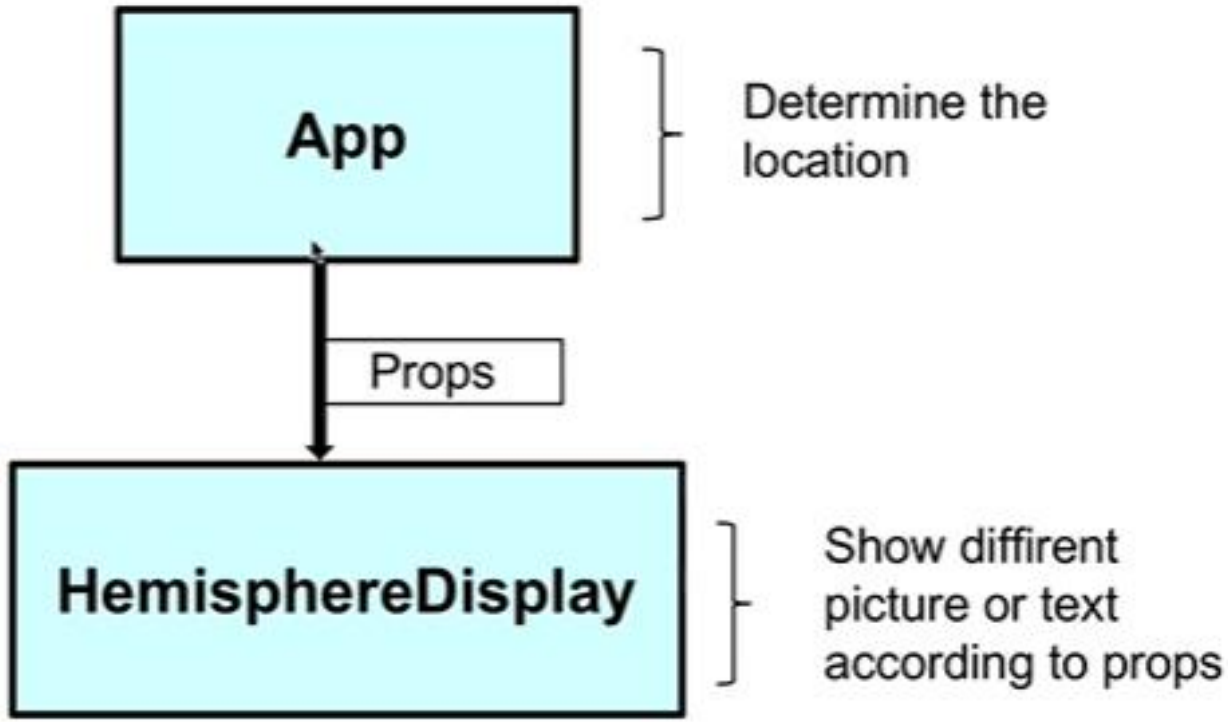


We can use it for everything else

Complex logic
Response to user input
Network request use
Class based component

Why class based component ?





Import React from ‘react’;
Import ReactDom ‘react-dom’;

Class based components

Takes time to get back information. The app component (function) will load before the data is transmitted, on possible condition of a call function.

Rules of Class based Components

- Must be a JavaScript Class
- Must extends React.Component
- Must define a ‘render’ method

index.js

```
1  import React from 'react';
2  import ReactDOM from 'react-dom';
3
4  class App extends React.Component {
5    render() {
6      window.navigator.geolocation.getCurrentPosition(
7        (position) => console.log(position),
8        (error) => console.log(error)
9      );
10     return(
11       <div>
12         You are in the northern hemisphere
13       </div>
14     )
15   }
16 }
17
18 ReactDOM.render(
19   <App />,
20   document.querySelector('#root')
21 )
```

State Rules

Only usable with class components

'State' is a JS object that contains data relevant to a component

Updating state on a component causes the component to rerender

State must be initialized when a component is created

State can only be updated using the function 'setState'

Rules of Class base components

- Must be a JavaScript class
- Must extends React.Component
- Must define a 'render' method

JS file loaded up by browser

App component gets created

Constructor function gets called

'this.state' property assigned the state object

We call geolocation service.

React calls the components render method

App returns JSX, gets rendered to page as HTML

index.js | index.html

```
1  import React from "react";
2  import ReactDOM from "react-dom";
3
4  class App extends React.Component {
5    constructor(props) {
6      super(props);
7      this.state = { latitude: null };
8      window.navigator.geolocation.getCurrentPosition(
9        position => {
10         this.setState({ latitude: position.coords.latitude });
11       },
12       error => console.log(error)
13     );
14   }
15   render() {
16     return (
17       <div>
18         Yes
19         <h1>TEST JSX</h1>
20         {this.state.latitude}
21       </div>
22     );
23   }
24 }
25
26 ReactDOM.render(<App />, document.querySelector("#root"));
27
```

supper(props)
this.state = {}
this.setState({})

We get results of geolocation!

We update our state object with a ‘this.setState’

React calls ‘render’ method second time

Render methods returns some JSX

React takes that JSX and updates content on the sceen.

Lifecycle Methods

constructor	→	Good place to one time setup
render	→	Just return JSX
componentDidMount	→	Good place to data loading
componentDidUpdate	→	Good place to do more data loading when state/props change
componentWillUnmount	→	Good place to do clean up