



#RahoAmbitious



Full Stack Software Development

Course: Advanced Frontend Development Using React

Lecture on: React Components

Instructor: Parth Dalal

TODAY'S AGENDA

- React Components
- Component visualization
- First React Component
- Types of components
- Custom Components
- Rendering content dynamically
- Composing and extracting components
- Smart components vs Dumb components

Poll 1

Name the file containing unit tests for the application

- A. index.js
- B. App.js
- C. App.test.js
- D. manifest.json



Poll 1 (Answer)

Name the file containing unit tests for the application

- A. index.js
- B. App.js
- C. **App.test.js**
- D. manifest.json



Poll 2

Which of the following can be the JavaScript expression to be evaluated inside curly braces during compilation? (Note: More than one option may be correct)

- A. A variable
- B. A function
- C. An object
- D. Any code snippet that returns some value



Poll 2 (Answer)

Which of the following can be the JavaScript expression to be evaluated inside curly braces during compilation? (Note: More than one option may be correct)

- A. **A variable**
- B. **A function**
- C. **An object**
- D. **Any code snippet that returns some value**



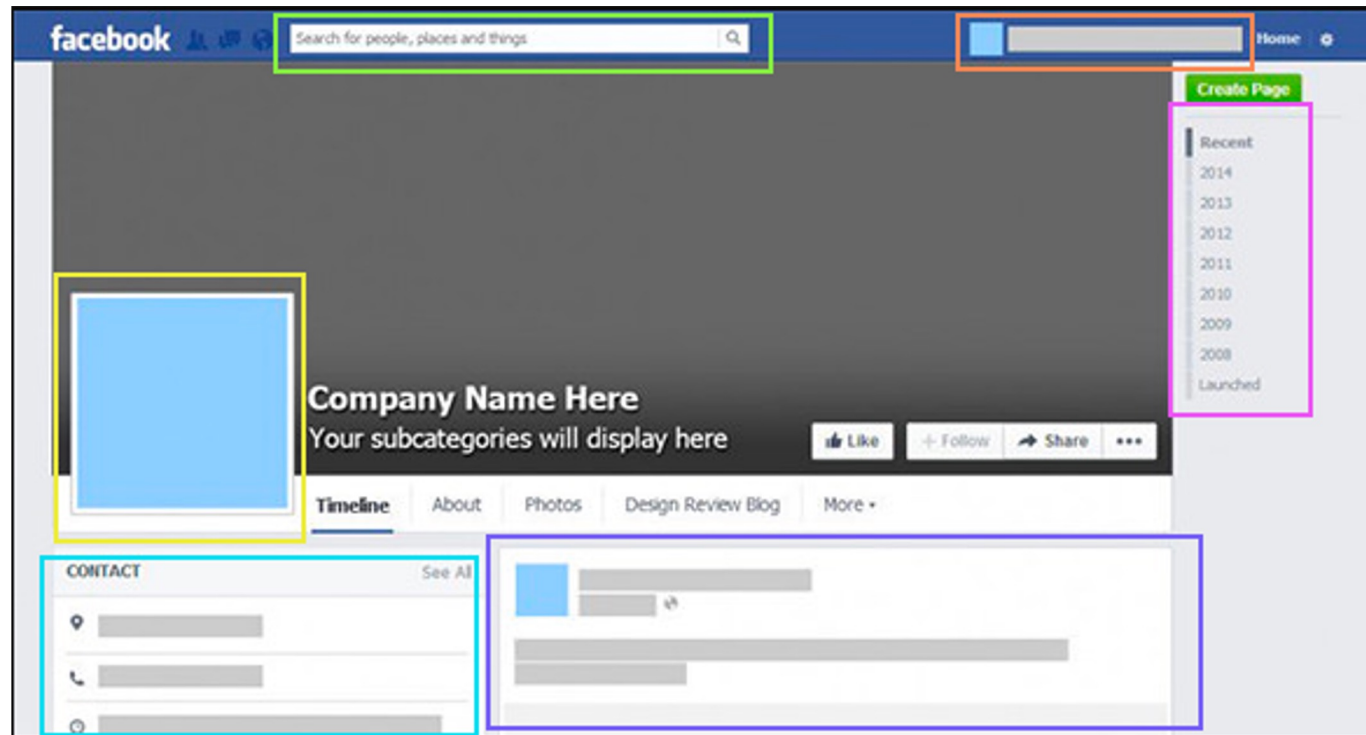
React Components

REACT COMPONENTS

- Components are the building blocks of React.
They help to break down large interfaces into small units that can be built and maintained independently.
- React follows a **component-based approach** where a component refers to an independent and reusable piece of code that can be plugged anytime anywhere
- Conceptually, components are like JavaScript functions
- They accept arbitrary inputs (called 'props') and return React elements describing what should appear on the screen
- Large application are bifurcated by these small components to make it more understandable and maintainable

REACT COMPONENTS

- Facebook uses more than 50,000 components, and they are all built using React. Here are some examples



REACT COMPONENTS

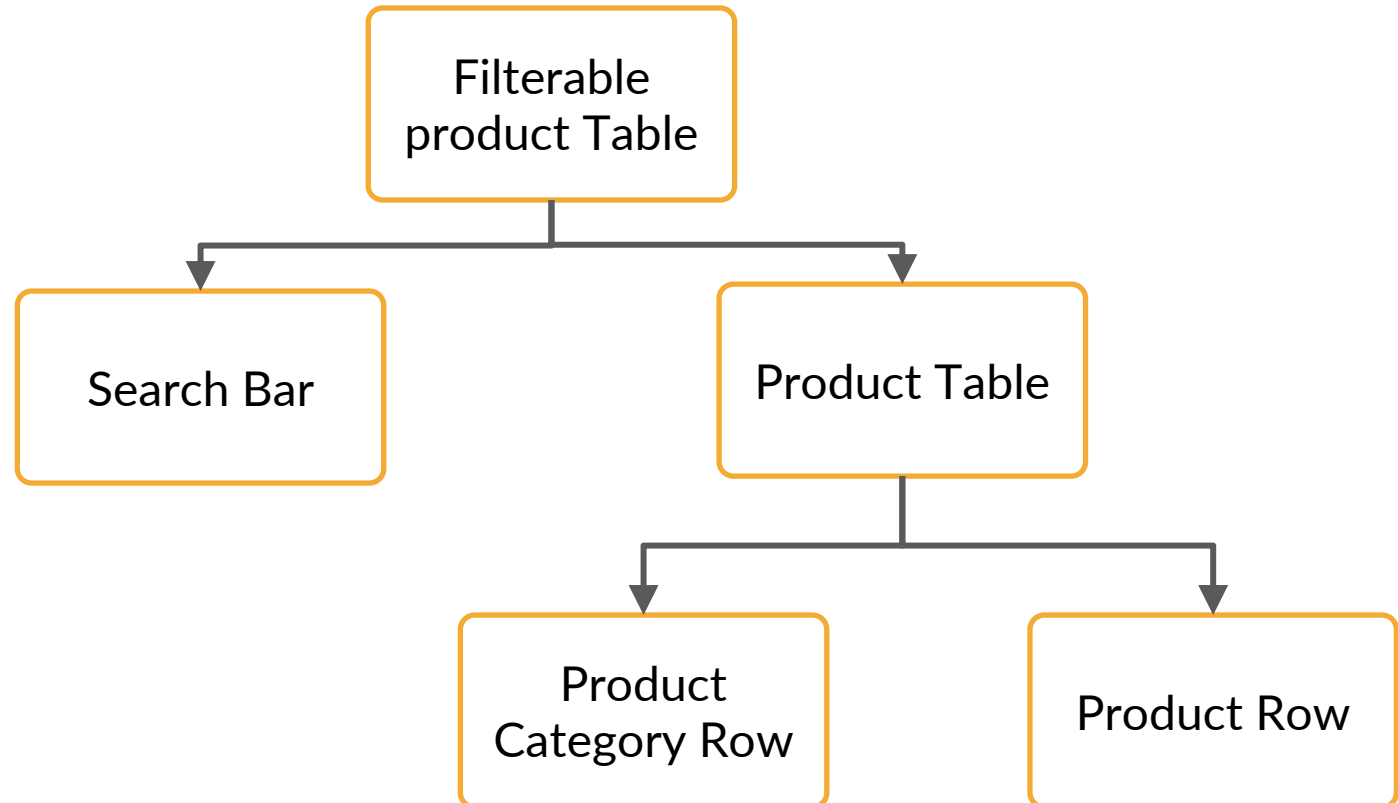
- Components help to break down monolithic codebases into small units.
- They are easy to maintain
- They can be developed independently as composable units
- They help to avoid logistical challenges during errors and debugging
- They bring scalability to the codebase
- Composable units of components when added with declarative rendering along with state and virtual DOM offer the perfect toolkit for building great UIs and applications

REACT COMPONENTS

Component Hierarchy Example

☐ Only show products in stock

Name	Price
Sporting Goods	
Football	\$49.99
Baseball	\$9.99
Basketball	\$29.99
Electronics	
iPod Touch	\$99.99
iPhone 5	\$399.99
Nexus 7	\$199.99



REACT COMPONENTS

Let's create the first React component

Open Requests

12

Closed Requests

2

Accepted Requests

10

Denied Requests

0

REACT COMPONENTS

- React is a unidirectional flow application
- Unidirectional flow is easy to understand and implement and can debug common problems
- Easy recognition of data sources and the flow

OVERVIEW OF THE APPLICATION

Here are some features of components:

- Components are reusable features that can be composed simultaneously to create a user interface
- The reusability of components allows you to design components that can be used across multiple projects, thereby saving time
- Modularising features in components facilitate scalability and maintainability

Types of Components

COMPONENTS

Types of Components

There are two types of components:

- **Functional components (stateless):** Written in the form of functions
- **Class components (stateful):** Written in the form of classes

Note: You will learn about state in the upcoming sessions

COMPONENTS

Types of Components

- A functional component is used when you need to pass data to a component and the component returns a React element
- On the other hand, class components provide additional features as compared to functional components
- A line of caution is that these class components, if not handled with care, can result in your application going into an inconsistent state

Note: Later in the course, you will discover that functional components can achieve whatever class components can and much more with help of hooks (a new addition in React 16.8)

Poll 3

Which of the following is/are the features of components in React?

- A. Self-contained
- B. Reusable
- C. Customisable
- D. All of the above



Poll 3 (Answer)

Which of the following is/are the features of components in React?

- A. Self-contained
- B. Reusable
- C. Customisable
- D. **All of the above**



COMPONENTS

Conditions of a Component

Here are some conditions to keep in mind while building components:

- A component's name must start with a capital letter in order to distinguish between the predefined HTML elements and the custom elements (components) created by users
For example, **MyComponent**
- A class component must extend from a base class named Component.
For this, you need to include a component as a named import from the 'react' library. If a component is not a named import, then you can extend from **React.Component**

COMPONENTS

Conditions of a Component

- A class component must always have the render() function

```
import React, {Component} from 'react';  
class MyComponent extends Component {  
    render() {  
        // code here  
    }  
}
```

COMPONENTS

- Render method in a component: Initial load of a component

```
class ToggleButton extends Component {  
  state = {  
    isOn: false  
  }  
  toggle = () => {  
    this.setState({  
      isOn: !this.state.isOn  
    });  
  }  
  render() {  
    return (  
      <div className={this.state.isOn ? "btn btn-on" : "btn"} onClick={this.toggle}>  
        {this.props.label}  
      </div>;  
    )  
  }  
}
```

The render method describes the visual elements of a component, typically using JSX

COMPONENTS

- State of a component: Stores values related to the component

```
class ToggleButton extends Component {
```

```
  state = {  
    isOn: false  
  }
```

State is used to describe visual elements using data. Updates to the state causes the component to re-render with visual updates.

```
  toggle = () => {  
    this.setState({  
      isOn: !this.state.isOn  
    });  
  }
```

```
  render() {  
    return (  
      <div className={this.state.isOn ? "btn btn-on" : "btn"} onClick={this.toggle}>  
        {this.props.label}  
      </div>;  
    )  
  }  
}
```


COMPONENTS

- Event listener in a component: Helps to perform additional functionalities like click

```
class ToggleButton extends Component {  
  state = {  
    isOn: false  
  }  
  toggle = () => {  
    this.setState({  
      isOn: !this.state.isOn  
    });  
  }  
  render() {  
    return (  
      <div className={this.state.isOn ? "btn btn-on" : "btn"} onClick={this.toggle}>  
        {this.props.label}  
      </div>;  
    )  
  }  
}
```

Listening to events such as mouse clicks



```
<div className={this.state.isOn ? "btn btn-on" : "btn"} onClick={this.toggle}>  
  {this.props.label}  
</div>;
```

COMPONENTS

Features of class components

- Class components let you store local state
- They allow you to create lifecycle hooks
- They have native support for state management
- They have instance methods
- They have lifecycle hooks

COMPONENTS

Features of functional components

- Functional components are simple Javascript functions that return a React element
- They can also bring in props object as an argument
- They are presentational components that accept data through a prop and renders it on the UI
- Functional components are perfect for separating visuals from container components

COMPONENTS

Class components vs functional components

- Unlike class components, function components do not offer features like built-in state management, lifecycle methods, etc. (**using React Hooks**)
- Considering their syntax and simplicity, function components are best-suited for elementary and simple use cases, unlike class components which are used for complex use cases

COMPONENTS

What should you pick between functional components and class components?

- Functional components coupled with the Hooks API offer a cleaner and simpler syntax
- They improve the readability of code and are also easier to test.

Note: There is nothing wrong with using class components. You will be using them throughout this course, along with function components

COMPONENTS

Custom component example

```
import React, { Component } from 'react';
import emoji from './emoji.svg'
export default class Instructions extends Component {
  render() {
    return(
      <>
        <img alt="laughing crying emoji" src={emoji} />
        <p>Click on an emoji to view the emoji short name.</p>
      </>
    )
  }
}
```

COMPONENTS

Here are some advantages of custom components:


- Make JavaScript coding easy
- Make template designing easy
- UI-focussed designs
- Easy to adopt
- Reusable

RENDER CONTENT DYNAMICALLY

- **Rendering data dynamically is an important aspect of React. One of the main examples of this is a list rendered dynamically using the map function**
- We will have an array of objects for which we will be using the JS map function to iterate over and print the values dynamically in an unordered list
- Without writing the HTML code multiple times, we will render data dynamically for it

RENDER CONTENT DYNAMICALLY

```
const numbers = [1, 2, 3, 4, 5];  
const listItems = numbers.map((number) =>  
  <li>{number}</li>  
);  
  
const root =  
ReactDOM.createRoot(document.getElementById('root'));  
root.render(<ul>{listItems}</ul>);
```

- 
- 1
 - 2
 - 3
 - 4
 - 5

COMPOSING AND EXTRACTION OF COMPONENTS

- Once you create a component, you can refer it in other components. By doing this, you can divide a component when it gets complex and create smaller components
- You can have a complete form component with smaller components like buttons and textboxes
- Here is an example of a component referring another component

```
function Pet(props) {  
  return <p>It's a {props.type}</p>;  
}  
function WhatsTheType() {  
  return (  
    <div>  
      <Pet type="Dog" />  
      <Pet type="Cat" />  
    </div>  
  );  
}  
ReactDOM.render(<WhatsTheType />,  
  document.getElementById("root"));
```

COMPOSING AND EXTRACTION OF COMPONENTS

- If you find your React component getting so big and complex that it's hard to keep track of everything, then you can divide it into smaller components for simplicity

```
function Dog(props) {  
  return (  
    <div className="dog">  
      <div className="dog-pic">  
        <img  
          className="avatar"  
          src={props.moreInfo.avatarUrl}  
          alt={props.moreInfo.alt}  
        />  
      </div>  
      <div className="dog-type">Type: {props.type}</div>  
      <div className="dog-name">Name: {props.name}</div>  
    </div>  
  );  
}  
  
const dog = {  
  name: "Rocky",  
  type: "puggle(?)",  
  moreInfo: {  
    avatarUrl: "https://picsum.photos/id/237/200/300",  
    alt: "dog image",  
  },  
};  
  
ReactDOM.render(  
  <Dog name={dog.name} type={dog.type} moreInfo={dog.moreInfo} />,  
  document.getElementById("root")  
);
```

SMART COMPONENTS VS DUMB COMPONENTS

- Smart components and dumb components are also known as container components and presentation components, respectively
- Smart components focus on how things work, whereas dumb components focus how things look
- Smart components manipulates data, call redux, lifecycle methods, APIs, libraries, etc. They also manage state and rarely include styling. Since dumb components focus on styling, smart components can focus on functionality without the clutter of styling
- Dumb components focus on the UI, accept props, require no app dependencies and rarely include state. These components are meant for presentation purpose only

SMART COMPONENTS VS DUMB COMPONENTS

- In the example below, almost every component can be a reusable dumb component including the container, header, inputs and button

The diagram illustrates a sign-in form with several nested rectangular boxes of different colors (yellow, teal, light blue) highlighting individual components that can be reused as 'dumb components'. The components include:

- Sign In** (Header)
- Username** (Label with a green checkmark icon)
- (Text input)
- That's a tasty looking email you've got there.* (Feedback message)
- Your username is most likely your email.* (Feedback message)
- Password** (Label with a green key icon)
- (Password input)
- Submit** (Button)

SMART COMPONENTS VS DUMB COMPONENTS

- Create a smart dropdown or a cascading dropdown, i.e., one dropdown value should depend on the previously selected one

KEY TAKEAWAYS

- Components are simply the JavaScript way of writing independent, reusable and dynamic code
- There are two types of components in React: functional components, which are written in the form of functions, and class components, which are written in the form of classes
- JavaScript's `map()` method can be used to iterate over an array and inject data into the React components or elements dynamically
- You need not hard-code data inside each component. This is one of the major reasons why React refers to its components as 'reusable' entities

KEY TAKEAWAYS

- Composing and extracting components
- Dumb components vs smart components

THE FOLLOWING TASKS ARE TO BE COMPLETED AFTER TODAY'S SESSION

MCQs
Coding Questions



Thank You!