upGrad

#RahoAmbitious

Full-Stack Software Development

upGrad

Course: Advanced Frontend Development Using React

Lecture on: Getting Started With React

Instructor: Mrigank Kaushik

In the last class, we discussed...

- Introduction to React
- History of React
- Advantages of React
- React environment
  - Role of Babel, Webpack and ESLint
- Single-page and multi-page applications and their advantages over one another

Poll 1

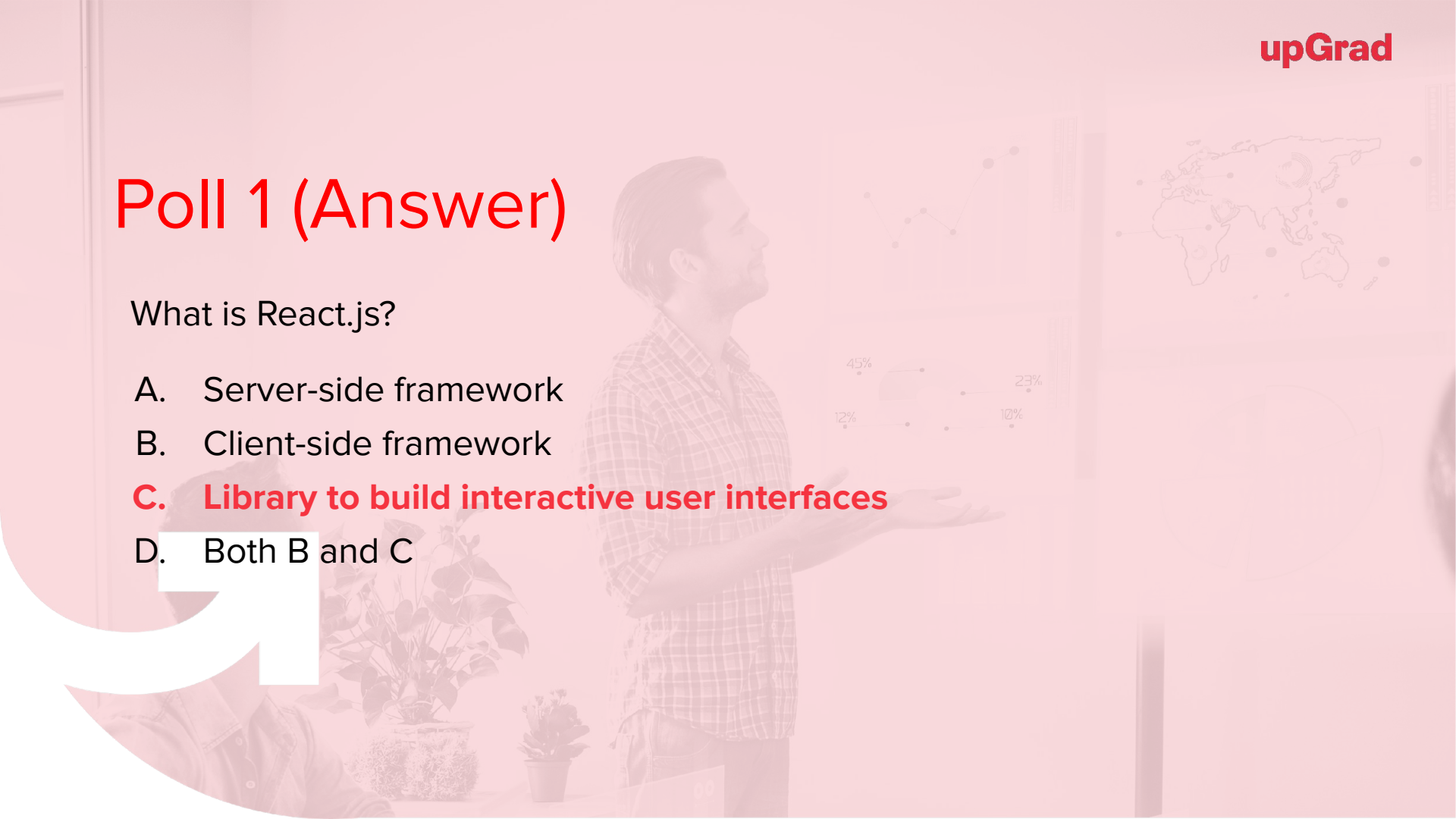
What is React.js?

A. Server-side framework
B. Client-side framework
C. Library to build interactive user interfaces
D. Both B and C

Poll 1 (Answer)

What is React.js?

A. Server-side framework
B. Client-side framework
C. Library to build interactive user interfaces
D. Both B and C



Poll 2

Fill in the blank.

React uses _____ for enhancing performance.

A. Real DOM
B. Virtual DOM
C. Both A and B
D. None of the above

Poll 2 (Answer)

Fill in the blank.

React uses _____ for enhancing performance.

A. Real DOM

B. Virtual DOM

C. Both A and B

D. None of the above

Today's Agenda

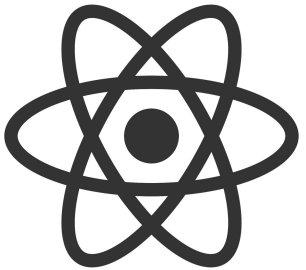
1. Overview of the Phone Directory application
2. How to set up the code
3. Folder structure and code cleanup
4. Introduction to JSX and how it is different from HTML
5. Injecting data using curly braces '{}' in React
6. The React.createElement() method
7. Rendering components in the root node
8. Rendering elements into DOM in React

Overview of the Application

Overview of the Application

upGrad

- It is time for you to learn more concepts of React by practically applying them
- To do so, you are going to build a Phone Directory application



Overview of the Application

upGrad

Phone Directory Application

- It is a mini phone directory that displays a list of subscribers
- It will have ‘Phone Directory’ as the header. The basic functionalities of the app will be addition and deletion of subscribers

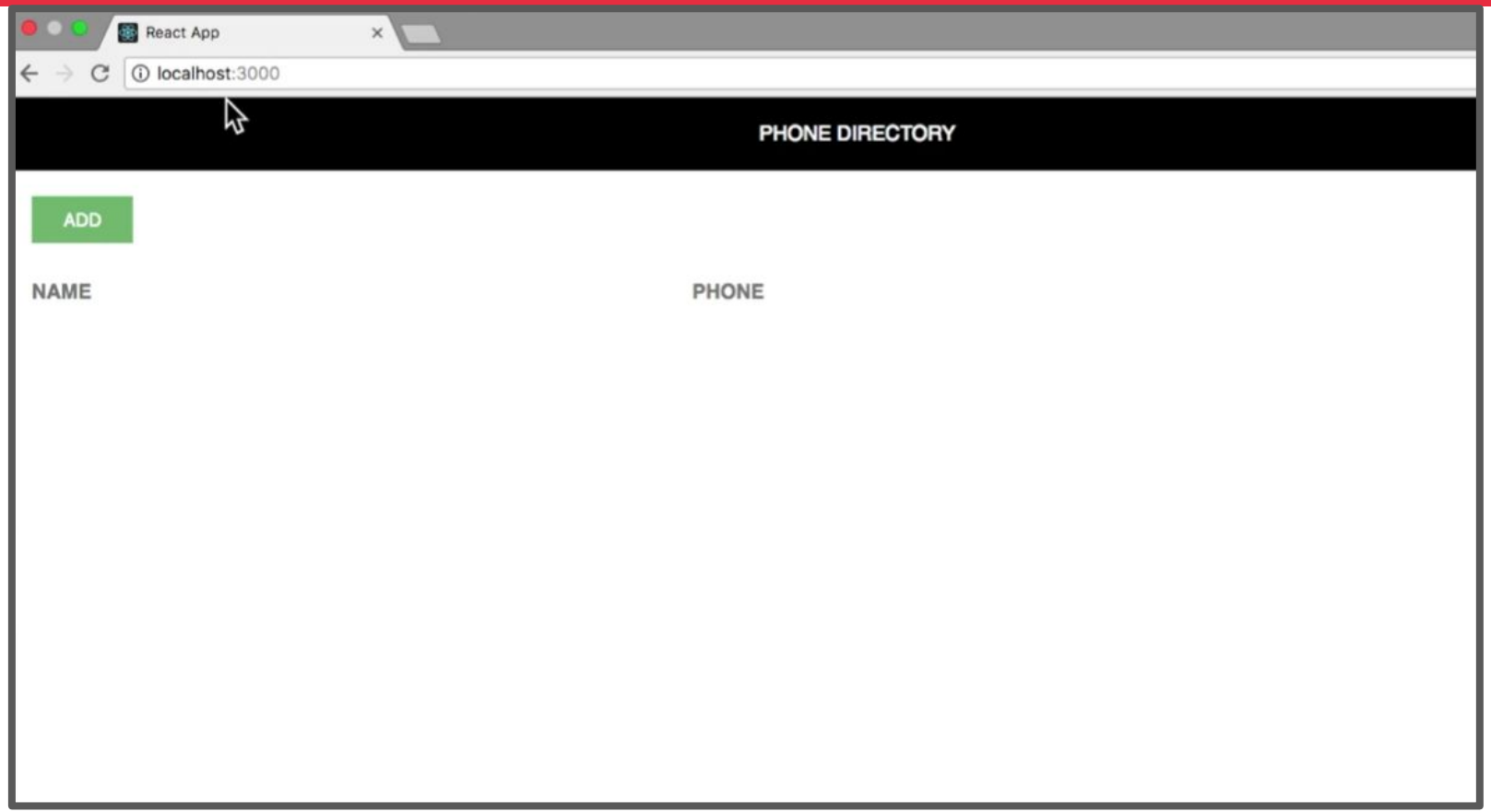
Overview of the Application

upGrad

- To add a subscriber, you will click on the ‘ADD’ button, which will direct you to the ‘Add Subscriber’ page, where you will be able to provide the name and phone number of the subscriber
- As you add the details, the state of the name and phone number will keep on changing below

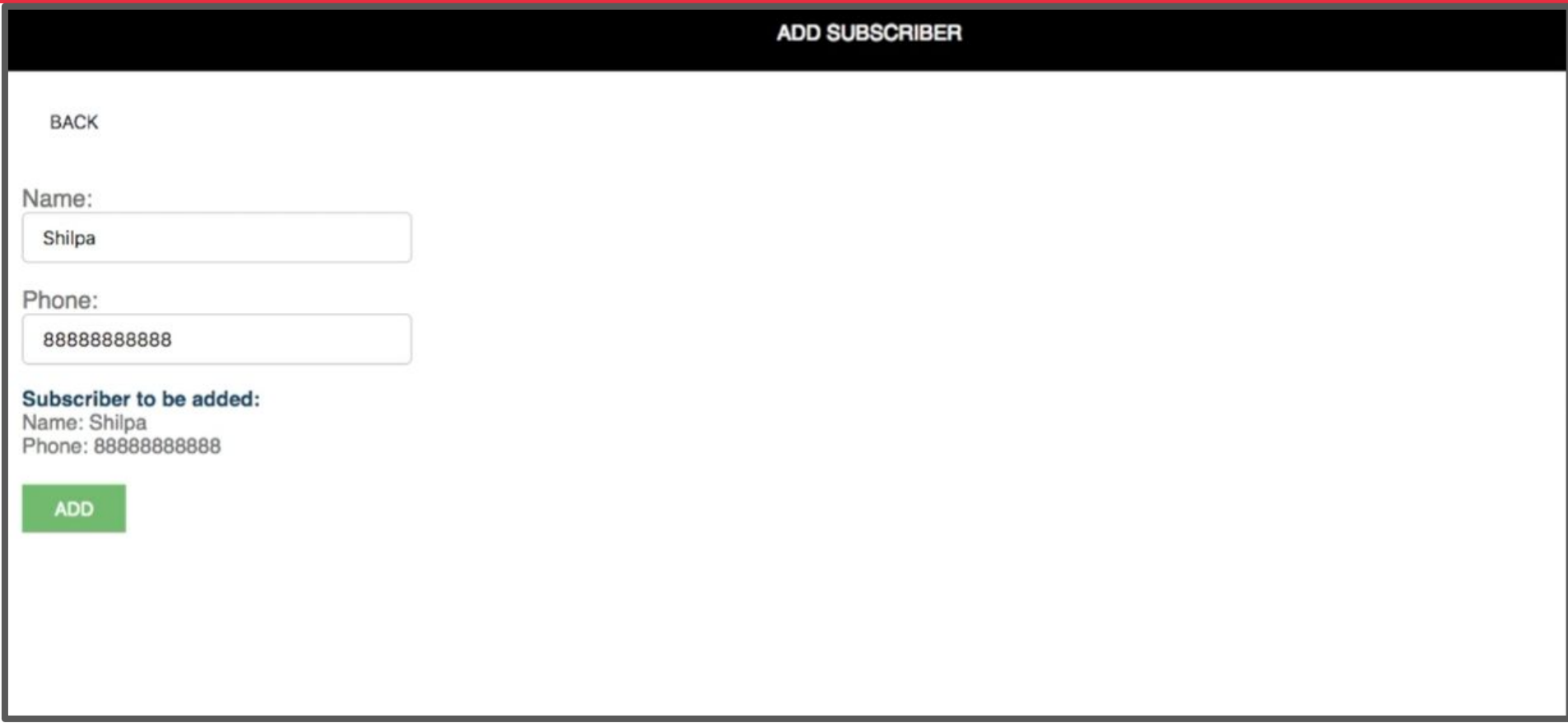
Overview of the Application

upGrad



Overview of the Application

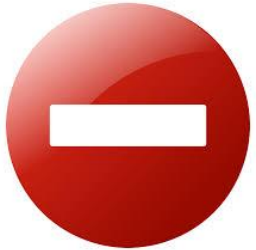
upGrad



Overview of the Application

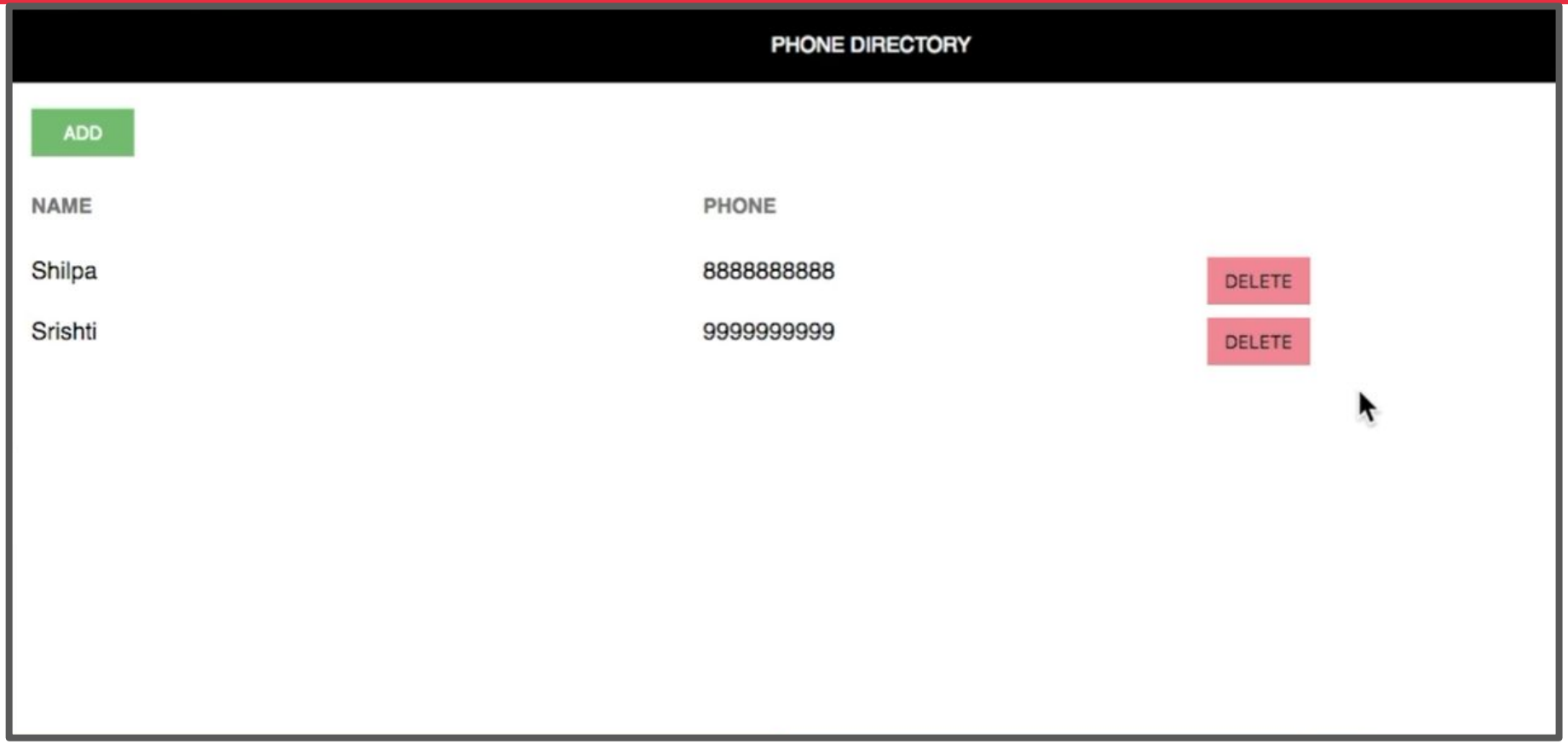
upGrad

- The subscribers will get added in line, one below the other
- There will also be a ‘DELETE’ button to delete subscribers from the directory



Overview of the Application

upGrad



Poll 3

According to you, what will happen when you click on the ‘Delete’ button to delete a subscriber's details on the homepage of the Phone Directory application?

A. The page will be reloaded, and the subscriber's details will be deleted

B. The page will not be reloaded, but the subscriber's details will be deleted

Poll 3 (Answer)

According to you, what will happen when you click on the ‘Delete’ button to delete a subscriber's details on the homepage of the Phone Directory application?

A. The page will be reloaded, and the subscriber's details will be deleted

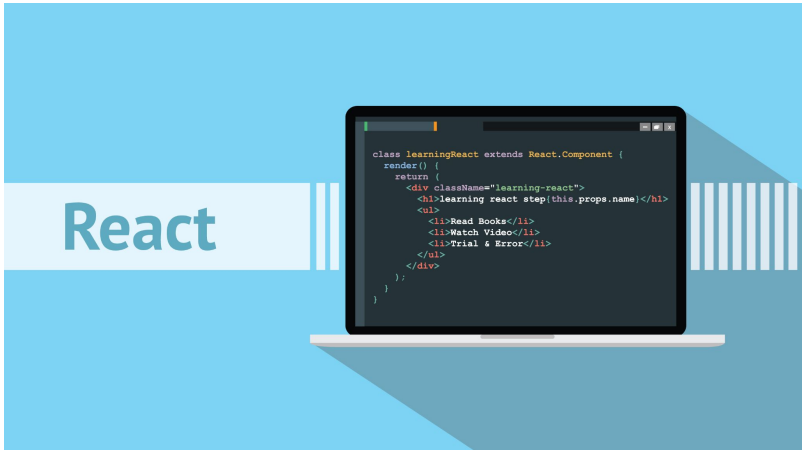
B. The page will not be reloaded, but the subscriber's details will be deleted

Codebase Set-up

Codebase Set-up

upGrad

Before starting to build the Phone Directory application, you need to set up your environment to create and run a React application



Codebase Set-up

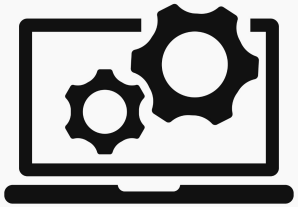
upGrad

Steps for Codebase Set-up

- Install Node.js
- NPM gets installed on your machine automatically while installing Node.js
- Use NPM to install the create-react-app package using the following command:

```
npm i -g create-react-app
```

To download Node.js, click here



Codebase Set-up

upGrad

- After you are done with the set-up, running command **`create-react-app phone-directory`** will create the folder structure
- You can see an application folder generated inside the path that you had chosen while creating the application. Inside this folder, you can see a bunch of files too, which have been created automatically
- This is nothing but your React application with the boilerplate configurations created by the create-react-app package by default so that you can focus only on the application logic

Poll 4

Which command is used to to install the create-react-app package?

A. `npm i -g create-react`

B. `npm i create-react app`

C. `npm create-react-app`

D. `npm i -g create-react-app`

Poll 4 (Answer)

Which command is used to to install the create-react-app package?

A. `npm i -g create-react`

B. `npm i create-react app`

C. `npm create-react-app`

D. `npm i -g create-react-app`

Understanding Visual Studio Code

Understanding Visual Studio Code

upGrad

- Before starting to build the application, you must have an overview of the files and folders inside your application folder
- To do this, you need to open the application folder inside a text editor, say **Visual Studio Code**

To download the Visual Studio Code, click here

Follow these steps to open your application folder inside the text editor:

1. Open Visual Studio Code
2. If you are on macOS, click on the ‘File’ menu at the top and then click on ‘Open...’. If you are using Windows, click on the ‘File’ menu at the top and then click on ‘Open Folder’
3. In the dialog box that opens, go to the location where your application folder resides and choose your application folder
4. In the application folder, choose the phone-directory folder
5. Click on ‘Select Folder’

Understanding Visual Studio Code

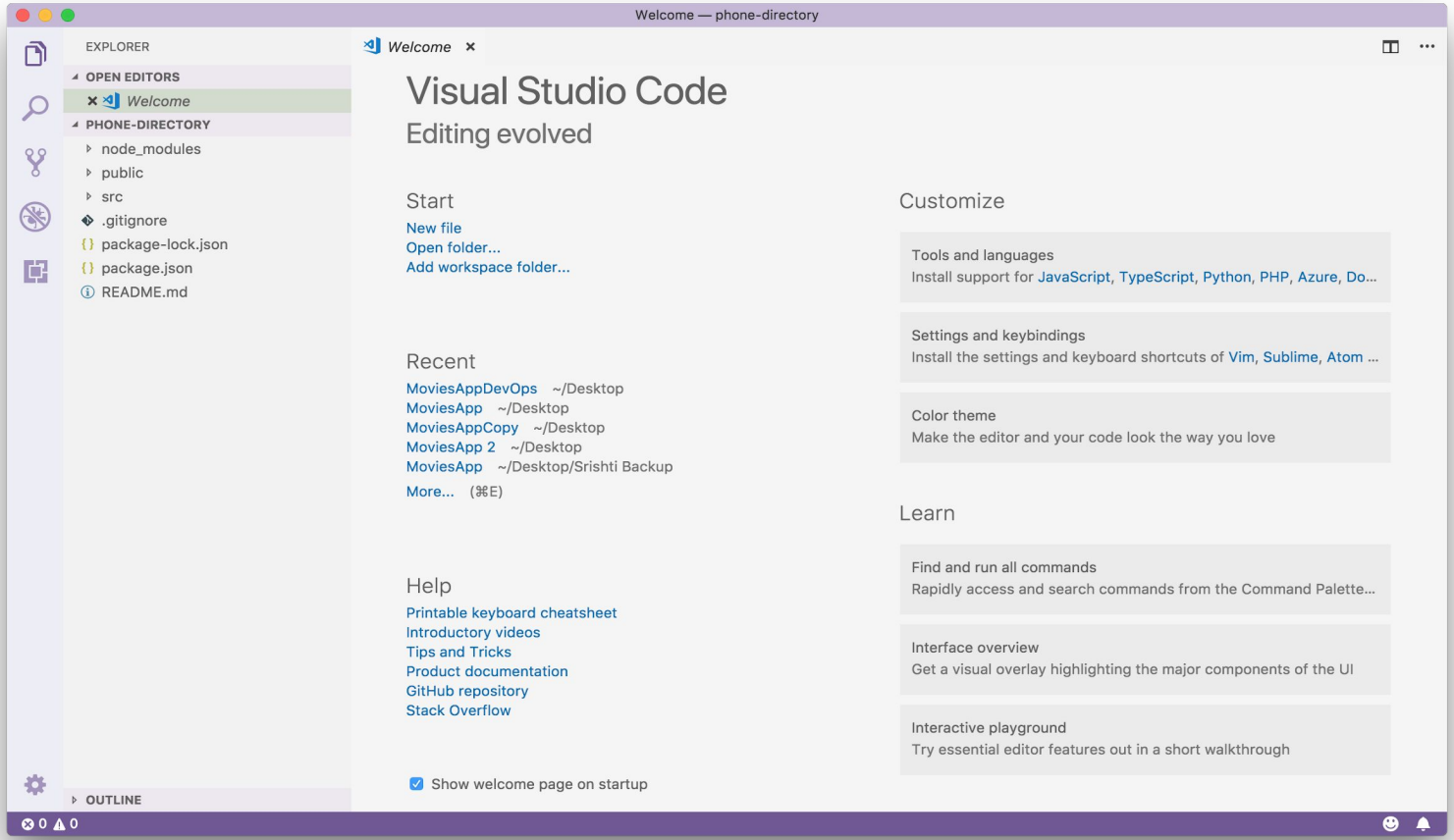
upGrad

When you look at the Explorer on the left side in Visual Studio Code, you can see the name of your application folder

You can also see a list of all the files and folders that are inside your application folder

Understanding Visual Studio Code

upGrad



Folder Structure

Folder Structure

upGrad

Following are the basic points to note about some files that you see inside your application folder:

.gitignore file

IGNORE USER

- It is used by Git to determine which files and directories to ignore before a commit is made
- It should be committed into the repository to share ignore rules with other users who clone the repository

Folder Structure

upGrad

- The *node_modules* folder is included inside the .gitignore file so that the user who clones the application is not required to clone this folder
- The user simply needs to run the command **`npm install`** in the root folder of the project
- This command creates the node_modules folder and installs all the dependencies (packages) needed for the application

Folder Structure

upGrad

package.json file

- It consists of the name and version of the application, the combination of which should be unique in order to publish the package
- It comprises dependencies that list all the packages needed to be installed for the application
- It also includes scripts that specify the commands to be run at various points in the application life cycle

Folder Structure

upGrad

package-lock.json file

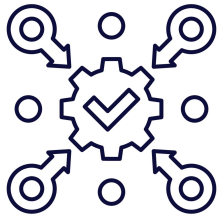
- It is automatically generated for any operation where npm modifies either the node_modules tree or the package.json file
- It locks the version of the full dependency tree of packages
- It guarantees the generation of an identical dependency tree when the application is cloned by other developers

Folder Structure

upGrad

node_modules folder

- Its contents are defined by the package.json file, and it consists of all the packages required for running your application



Folder Structure

upGrad

public folder

- Nothing inside this folder is processed by webpack
- It is used for storing small files that are not required to be bundled
- It can be used to store images when they are in thousands and their paths need to be referenced dynamically
- Any file inside this folder needs to be referenced at other places using the %PUBLIC_URL%/ keyword, which gets replaced with the path of the public folder during the application's build process

Folder Structure

upGrad

index.html file

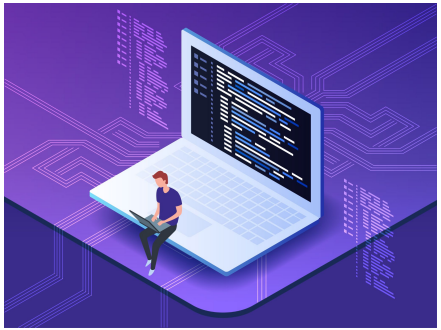
- It is the starting point of the application
- It should always remain with the name index.html and inside the public folder; otherwise, the code will fail to run
- It can only reference files that are inside the public folder

Folder Structure

upGrad

manifest.json file

- It is a simple JSON file telling the browser about the web application and its behaviour when it is installed on the user's mobile device or computer



Folder Structure

upGrad

src folder

- It consists of the real application code
- It consists of all the files that are needed to get bundled by webpack

Folder Structure

upGrad

index.js file

- It is the entry point for JavaScript
- The filename should remain as index.js, and the location should be inside the src folder; otherwise, the code will not run

Folder Structure

upGrad

index.css file

- It is the style sheet for index.html

Folder Structure

upGrad

registerServiceWorker.js file

- It is the web browser API that is used to cache assets and other files to work passively in the background. It helps offline users or the ones who are on a slow network to see results on the screen faster
- It adds offline capabilities to the application

Folder Structure

upGrad

App.js file

- It is the JavaScript file for the App.

App.css file

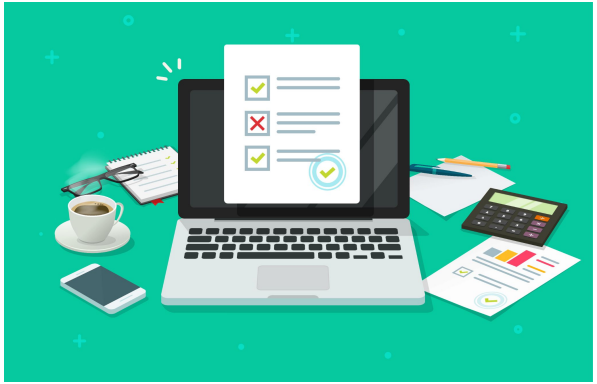
- It is the style sheet for the App.

Folder Structure

upGrad

App.test.js file

- It is the test file for the App
- It contains unit tests for the application
- It runs test cases for all the files that changed since the last commit of the application



upGrad

logo.svg file

- SVG is an acronym for Scalable Vector Graphics
- An SVG file is an XML-based vector image format for 2D graphics, with support for interactivity and animation
- It is similar to raster-based image formats, such as JPEG, PNG, BMP, GIF, etc.

- It offers a bandwidth-friendly way of rendering images; no matter how large a graphic gets, it transmits only the XML describing the graphic to the client
- It helps to render resolution-independent and SEO-friendly images

Folder Structure

upGrad

As you can see, a lot of files are created automatically for you when you create a React application using the *create-react-app* node package

Code Reference

Poll 5

Which file is considered the starting point of the application?

A. index.html
B. package.json
C. index.js
D. App.js

Poll 5 (Answer)

Which file is considered the starting point of the application?

A. index.html

B. package.json

C. index.js

D. App.js

Poll 6

Which folder consists of all the files that need to get bundled by webpack?

A. node_modules

B. src

C. public

Poll 6 (Answer)

Which folder consists of all the files that need to get bundled by webpack?

A. node_modules

B. src

C. public

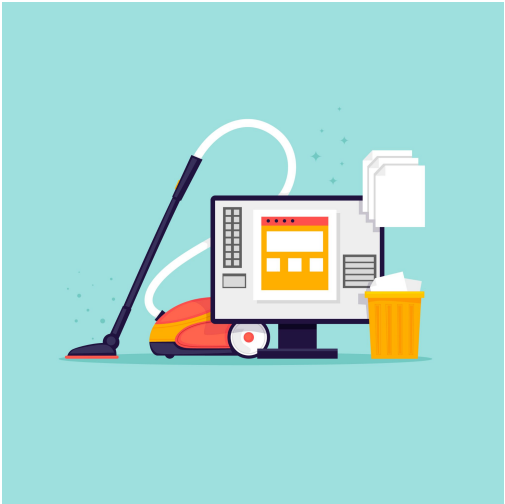
Code Clean-up

Code Clean-up

upGrad

Since you will be writing your own custom code, you do not need all the auto-generated code and files in your application

So, let's clean up some unnecessary code before you move on to building the application



Code Clean-up

upGrad

Follow the steps mentioned below to remove the unwanted code and files from your application:

- Inside the src folder, go to the *App.js* file
  - Remove all the code written inside the div of the return statement of the *App* function
  - Remove the *className="App"* from the outer div while leaving the outer div as it is
  - Remove the import statements at the top of the file for the *logo.svg* and *App.css* files
- Inside the *src* folder, delete the *App.css* and *logo.svg* files

Code Clean-up

upGrad

After the code clean-up, the *App.js* file will look like this:

```
function App() {
  return (
    <div>
    </div>
  );
}

export default App;
```

Code Reference

Introduction to JSX

Introduction to JSX

upGrad

The *div* element in the previous slide looks like HTML but is actually a syntax called JSX used by React.

So, what is JSX?

- JSX is a syntax extension to JavaScript created by Facebook by combining JavaScript and their proprietary markup language, XHP (combination of XML and PHP)
- Thus, in a React application, JSX gets converted into JavaScript and produces React elements
- Though JSX is an HTML-looking syntax, it is actually an XML extension to the ECMAScript specification. Thus, instead of using pure JavaScript for building DOM elements, you can use JSX, which offers flexibility to developers to use a familiar syntax, HTML

Introduction to JSX

upGrad

Let us start with adding a header to our Phone Directory Application.

- Go to App.js
- Remove the outer div
- Write ‘Phone Directory’ which is the heading of our app inside return method

```
function App() {
    return (
      <div>
        Phone Directory
      </div>
    );
}
```

Code Reference

JSX vs HTML

Now that you already know about a syntax similar to HTML named JSX, it is time to learn about the differences between HTML and JSX

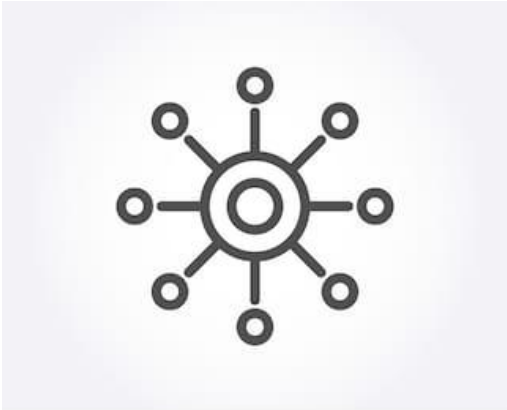
VS

JSX vs HTML

- **Adjacent JSX elements must be wrapped in an enclosing tag**
  - While returning JSX from a function or a class, you cannot return multiple elements
  - You can return only a single element
  - This is the reason why you need to encompass all children elements within a parent element and then, return this parent element
  - In case you fail to do this, you will get a syntax error saying, ‘Adjacent JSX elements must be wrapped in an enclosing tag’

JSX vs HTML

- However, in HTML, you can return as many DOM elements as you want
- You do not have any rule of returning a single element



JSX vs HTML

- Note that with the introduction to React 16, one can return an array consisting of multiple elements existing at the same level
- These elements are separated from each other using a comma

JSX vs HTML

- Thus, one can write the following code snippet, which works fine in React 16:

```
return [
    <div>
        Phone Directory
    </div>,
    <button>Add</button>,
    <div>
        <span>Name</span>
        <span>Phone</span>
    </div>
]
```

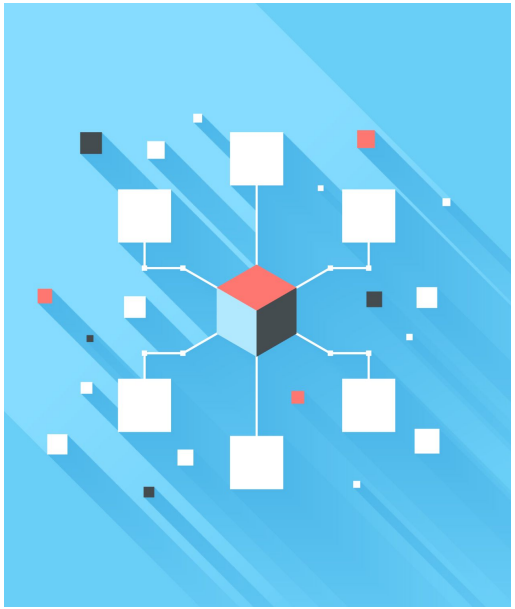
- However, when React released its v16.2, it introduced Fragment, which allows you to return multiple elements. You can reference Fragment as mentioned below:

```
import { Fragment } from 'react';
```

- After including Fragment as a named import from the ‘react’ library, you can wrap multiple elements inside it
- At last, the entire Fragment enclosing multiple elements can be returned

- Thus, you can simply write the following code snippet:

```
return (
    <Fragment>
        <div>
            Phone Directory
        </div>
        <button>Add</button>
        <div>
            <span>Name</span>
            <span>Phone</span>
        </div>
    </Fragment>
)
```



JSX vs HTML

- **Closing tag required**
  - In JSX, you need to close both types of tags: opening-closing tags as well as self-closing tags
  - For an opening tag, you need to explicitly write a closing tag at the end
  - For a self-closing tag, you need to put a forward slash before the closing angular bracket
  - If you fail to do this, you will get a syntax error saying, ‘Expected corresponding JSX closing tag for
’
  - However, in HTML, the browser sometimes takes care of closing the tags by itself

JSX vs HTML

- **JSX properties are not similar to HTML attributes**
  - Some attributes that you use in HTML cannot be used as JSX properties
  - This is due to the reason that all of the JSX code gets converted to JavaScript code at the end
  - You know that JavaScript has its own set of keywords
  - If you try to write these keywords or reserved words as JSX properties, it gets confusing to identify when the word is being used as a JavaScript keyword (or reserved word) and when it is being used as a JSX property

JSX vs HTML

- To make this distinction, use alternative keywords in JSX for those HTML attributes that exist in JavaScript language
- Examples:
  - **`htmlFor`** is used in React instead of **`for`**
  - **`dangerouslySetInnerHTML`** is used in React instead of **`innerHTML`**
  - Use the **`defaultValue`** or **`value`** props on <select> instead of setting **`selected`** on <option>.

- **Case sensitiveness**
  - React ‘reacts’ to cases that you use
  - It does not allow you to write something in any case of your choice
  - On the other hand, HTML syntax is not case sensitive. You can choose to write the div tag as <DIV>, or <Div>

Let us now create the skeleton of the Home page in the Phone Directory application using JSX.

```
<div>
    Phone Directory
    <div className="header">
            Phone Directory
    </div>
    <button>Add</button>
    <div>
        <span>Name</span><br />
        <span>Phone</span>
    </div>
</div>
```

Code Reference

Poll 7

Which of the following is a distinguishing characteristic between JSX and HTML?
(Note: More than one option may be correct.)

A. Adjacent elements need not be wrapped in an enclosing tag
B. Closing tag is required for all types of tags
C. JSX properties are similar to HTML attributes
D. Case sensitiveness of the syntax

Poll 7 (Answer)

Which of the following is a distinguishing characteristic between JSX and HTML?
(Note: More than one option may be correct.)

A. Adjacent elements need not be wrapped in an enclosing tag

B. Closing tag is required for all types of tags

C. JSX properties are similar to HTML attributes

D. Case sensitiveness of the syntax

Injecting Data Using Curly Braces {} in React

Injecting Data Using Curly Braces {} in React

upGrad

“Hi Srishti, Thanks for placing your order!”

Have you ever got an email like this after placing an order on an e-commerce website?

Injecting Data Using Curly Braces {} in React

upGrad

What do you think of it? Do you think that they have an executive who sends customised emails to each and every customer?

Well, if that happens to be the case, this is an excessive amount of work when there are tens of thousands of customers placing orders every second.

Injecting Data Using Curly Braces {} in React

upGrad

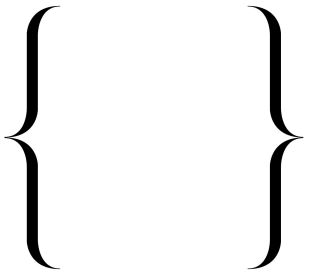
So, what do they do then?

You may have thought of the solution by now. They must have some sort of an email template that consists of common data for all customers.

Injecting Data Using Curly Braces {} in React

upGrad

- Well, this can be done in React using curly braces {}. This can be used to evaluate a JavaScript expression during compilation
- The expression can be a variable, a function, an object, an arithmetic calculation, logical evaluation, or any code snippet that returns some value



Injecting Data Using Curly Braces {} in React

upGrad

Example

```
let moduleName="React";
<span>Learning {moduleName} is so much fun!</span>
```

Output

Learning React is so much fun!

Injecting Data Using Curly Braces {} in React

upGrad

Also, JavaScript code can be written below the *render* method and above the *return* without using curly braces:

```
render() {
    const buttonStyle = {};
    if(this.state.toggleOn){
        buttonStyle.backgroundColor = "red";
    }
    else {
        buttonStyle.backgroundColor = "blue";
    }

    return (
        <button style = {buttonStyle} onClick={this.handleClick}>
            {this.state.toggleOn ? 'ON' : 'OFF'}
        </button>
    );
}
```

React.createElement() method

React.createElement() Method

upGrad

Let’s see how we can convert JSX to JavaScript

Well, this is done using the **React.createElement()** method

React.createElement() Method

upGrad

Syntax:

```
React.createElement(element_name, element_properties,
children);
```

React.createElement() Method

upGrad

- The first argument in this method is the name of the element that is to be rendered
- This can be either your custom component or an HTML element
- The second argument is the object that consists of property-value pairs that can be provided as attributes to this component or element
- After these two arguments, you can pass an infinite number of children elements, which will be nested inside this main component or element
- These children can, in turn, have other children elements nested inside them
- The first argument is mandatory, while the rest of the arguments that follow are optional

React.createElement() Method

upGrad

Example

JSX Code:

```
<div id="module">
   <p>ReactJS</p>
</div>
```

JavaScript Code:

```
React.createElement("div", {id:
"module"},
   React.createElement("p",
null, "ReactJS")
);
```

React.createElement() Method

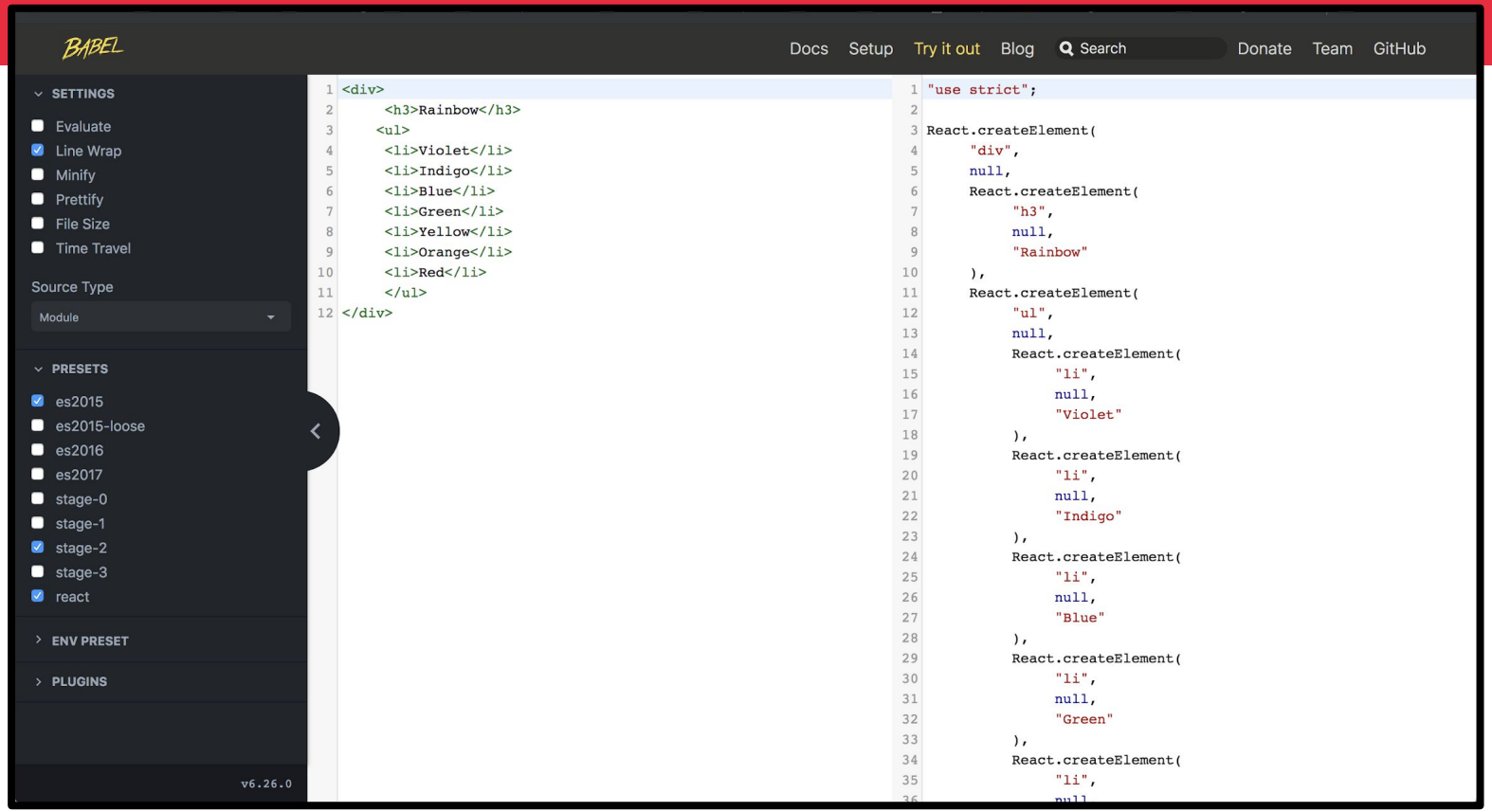
upGrad

In the Phone Directory application, when you write the code in the previous example, you will observe that both the code snippets in the last slide do the exact same thing.

When you try to write the aforementioned JSX code, Babel converts it into JavaScript code as shown in the example.

React.createElement() Method

upGrad



Fewer lines of JSX code getting converted to a lot of lines of JavaScript code

React.createElement() Method

upGrad

Note: A JSX expression must have one parent element, else it gives two return statements

```
render() {
    return (
        <h1>Welcome to React</h1>
        <div>Hi</div>
    );

    return React.createElement("h1", null,"Welcome to React");
    return React.createElement(div, null,"Hi");

}
```

Poll 8

Which method do you use to create a new element on DOM? Mention the syntax.

A. `React.createElement(element_name, element_properties, parent);`

B. `React.createElement(element_properties, element_name, children);`

C. `React.createElement(element_name, children);`

D. `React.createElement(element_name, element_properties, children);`

Poll 8 (Answer)

Which method do you use to create a new element on DOM? Mention the syntax.

A. `React.createElement(element_name, element_properties, parent);`

B. `React.createElement(element_properties, element_name, children);`

C. `React.createElement(element_name, children);`

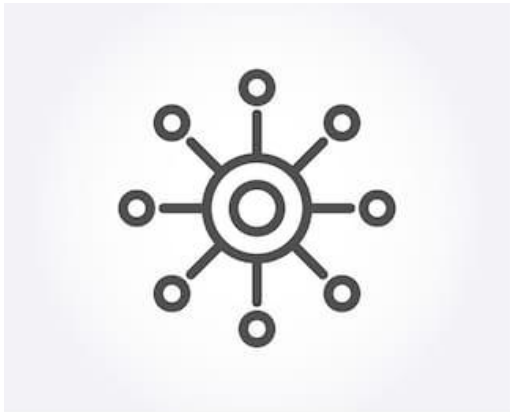
D. `React.createElement(element_name, element_properties, children);`

Rendering Components in Root Node

Rendering Components in Root Node

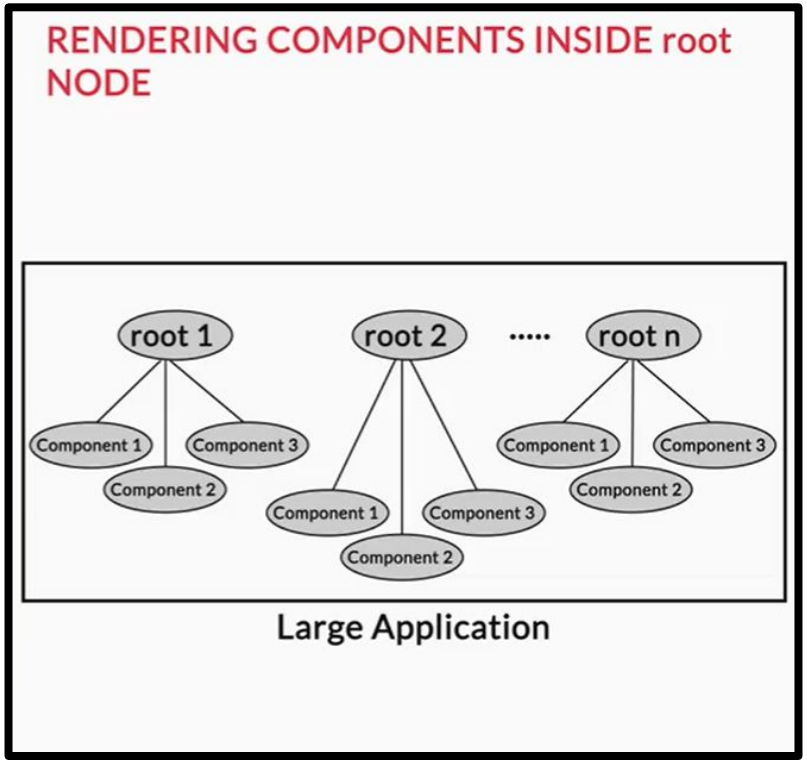
upGrad

- You know that you have one root node inside your application that contains all of the React code
- But, can you have multiple root nodes inside one application?
- Well, yes, you can have as many root nodes inside your application as you want
- One root node suffices in a small application, but you can have **multiple root nodes** depending upon the needs of your application



Rendering Components in Root Node

upGrad



Rendering Components in Root Node

upGrad

When do you need to have multiple root nodes inside an application?

- One of the advantages of using React is that it does not make any assumption about your technology stack
- Imagine that you want to ship only some features in React in an application that is not primarily built using React
- These features are spread across your entire application and written at multiple places inside the application
- Now, you need to achieve this without touching the rest of your independently existing code

Rendering Components in Root Node

upGrad

- This can be possible only when you are able to choose selective pieces of your existing code to be shipped, convert them to React, and then plug these back into different places inside your application
- Having multiple root elements help you achieve this and, thus, redefine your application using React
- Each root node can contain multiple React components, and these root nodes can be plugged into different places inside your application

Rendering Elements Into DOM

Rendering Elements Into DOM

upGrad

How do you render elements into DOM in React?

You can use the **`ReactDOM.render()`** method for this purpose

```
ReactDOM.render(<App/>, document.getElementById('root'));
```

Syntax

```
ReactDOM.render(argument_1, argument_2);
```

Rendering Elements Into DOM

upGrad

- There must be two arguments in the ReactDOM.render() method
- The first argument of the ReactDOM.render() method tells you what is to be rendered
    - This does not mean that only one element can be rendered in the first argument
    - Multiple elements can be rendered by enclosing them inside a parent div container
    - Also, this argument does not necessarily have to be a component
    - This argument can also take JSX code directly
- The second argument of the ReactDOM.render() method can be anything that specifies a location on the DOM, where the elements in the first argument need to be rendered

Poll 9

Which of the following is true with respect to the arguments of the ReactDOM.render() method?

```
ReactDOM.render(argument_1, argument_2);
```

(Note: More than one option may be correct.)

A. argument_1 tells WHAT to render.
B. argument_2 tells WHERE to render.
C. *argument_1* tells WHERE to render.
D. *argument_2* tells WHAT to render.

Poll 9 (Answer)

Which of the following is true with respect to the arguments of the ReactDOM.render() method?

```
ReactDOM.render(argument_1, argument_2);
```

(Note: More than one option may be correct.)

A. argument_1 tells WHAT to render.

B. argument_2 tells WHERE to render.

C. *argument_1* tells WHERE to render.

D. *argument_2* tells WHAT to render.

All the code used in today's session can be found in the link provided below (branch session2-demo):

https://github.com/upgrad-edu/react-class-components/tree/session2-demo

Doubt Clearance (5 minutes)

The following tasks are to be completed after today's session

MCQs
Coding Questions
Course Project (Part A) - Checkpoint 1

Key Takeaways

- Using the *create-react-app* package, you created your React application, Phone Directory. You started your development server on your local machine to run this application
- You also looked at which files and folders are automatically created by the *create-react-app* package and what they are meant for
- You deleted the unnecessary code from the auto-generated application folder
- You learnt about JSX, which is an HTML-looking syntax but is actually an XML extension to ECMAScript specification. Thus, instead of using pure JavaScript for building DOM elements, you can use JSX, which offers flexibility to developers to use a familiar syntax, HTML

Key Takeaways

- You understood how the following points hold true for React but not for HTML:
  - Adjacent JSX elements must be wrapped in an enclosing tag
  - Closing tag required for all types of tags
  - JSX properties are not similar to HTML attributes
  - Case sensitiveness of the syntax
- Curly braces can be used in React to evaluate a JavaScript expression during compilation. The expression can be a variable, a function, **an object,** an arithmetic calculation, logical evaluation or any code snippet that returns some value
- React.createElement() and ReactDOM.render() methods are used when an element needs to be created or rendered into DOM, respectively

In the next class, we will discuss...

1. Components, their types and conditions
2. Inline styling in React
3. External styling in React
4. Dynamic rendering using the map() method

upGrad

#RahoAmbitious

Thank you!