ReactJS

June 2021

Course Objective

- React Introduction
- Environment Setup
- ES6
- React Render Html
- JSX
- React Components
- Props and State
- Component API
- Component Life cycle
- React Forms, Events Refs, Keys
- Rest API
- React Router

Session Plan

- React Forms
- Events Refs
- Keys
- Rest API
- React Router

Forms

React uses forms to allow users to interact with the web page.

There are mainly two types of form input in React.

- Uncontrolled component
- Controlled component

Uncontrolled Components

- The uncontrolled input is like the traditional HTML form inputs.
- The DOM handles the form data, and the HTML elements maintain their own state that will be updated when the input value changes.
- ref is used to get the form values in Uncontrolled Component.

Uncontrolled Component - form

```
constructor(props) {
    super(props);
    this.updateSubmit =
this.updateSubmit.bind(this);
    this.input = React.createRef();
 }



 updateSubmit(event) {
    alert('You have entered the UserName
and CompanyName successfully.');
    event.preventDefault();
 }
```

```
<form onSubmit={this.updateSubmit}>
      <h1>Uncontrolled Form Example</h1>
      <label>Name:
        <input type="text" ref={this.input} />
      </label>
      <label>
        CompanyName:
        <input type="text" ref={this.input} />
      </label>
      <input type="submit" value="Submit" />
    </form>
  );
 }
```


Controlled Component

- In HTML, form elements maintain their state and update it according to the user input.
- In the controlled component, the input form element is handled by the component rather than the DOM.
- A controlled component takes its current value through props and notifies the changes through callbacks like an onChange event.
- A parent component "controls" this changes by handling the callback and managing its own state and then passing the new values as props to the controlled component.

Controller Component

```
<form onSubmit={this.handleSubmit}>
          <h1>Controlled Form Example</h1>
          <label>
             Name:
             <input type="text"
value={this.state.value}
onChange={this.handleChange} />
          </label>
          <input type="submit" value="Submit" />
        </form>
      )
```

```
constructor(props) {
      super(props);
      this.state = {value: ''};
      this.handleChange = this.handleChange.bind(this);
      this.handleSubmit = this.handleSubmit.bind(this);
  }
```

```
handleChange(event) {
      this.setState({value: event.target.value});
  }
```

```
handleSubmit(event) {
      alert('You have submitted the input successfully: ' +
this.state.value);
      event.preventDefault();
  }
```

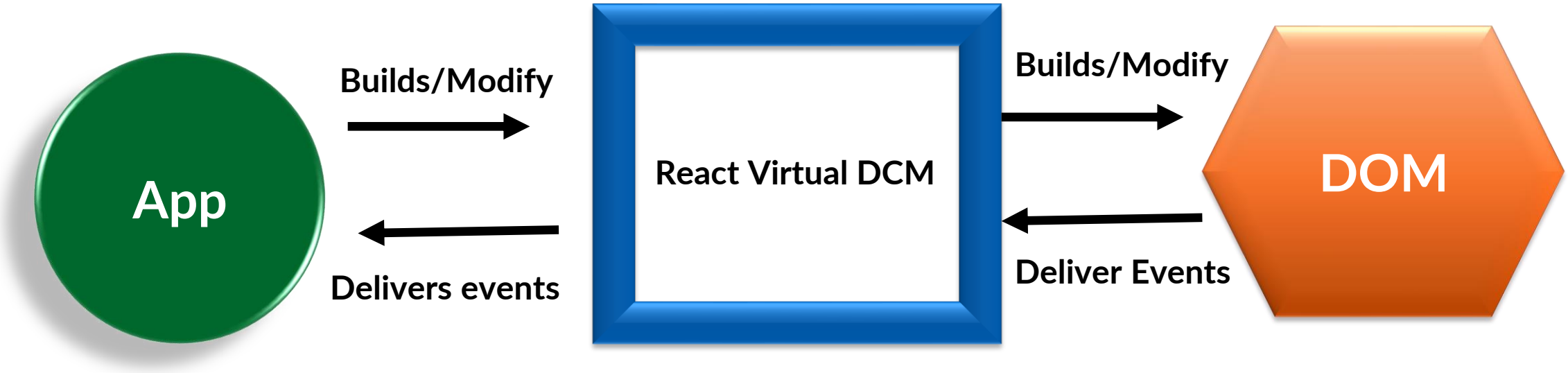
Controlled Vs Uncontrolled Component

SN	Controlled	Uncontrolled
1.	It does not maintain its internal state.	It maintains its internal states.
2.	Here, data is controlled by the parent component.	Here, data is controlled by the DOM itself.
3.	It accepts its current value as a prop.	It uses a ref for their current values.
4.	It allows validation control.	It does not allow validation control.
5.	It has better control over the form elements and data.	It has limited control over the form elements and data.

React Events

- An event is an action that could be triggered as a result of the user action or system generated event.
- React can perform actions based on user events like click, change, mouseover etc.
- React has its own event handling system which is very similar to handling events on DOM elements.

Event Handler



Event handler

```
render() {
      return (
        <div>
          <button onClick =
{this.updateState}>CLICK</button>
          <h4>{this.state.data}</h4>
        </div>
      );
```

```
 this.state = {
          data: 'Initial data...'
      }
      this.updateState =
this.updateState.bind(this);

   };
```

```
updateState() {
    this.setState({data: 'Data updated...'})
  }
```

Child Event

We have to create the event handler in the parent component and send the data as prop to the child component.

```
constructor(props) {
    super(props);

    this.state = {
      data: 'Initial data...'
    }
    this.updateState = this.updateState.bind(this);
  };
```

```
render() {
    return (
      <div>
        <Content myDataProp = {this.state.data}
            updateStateProp = {this.updateState}></Content>
      </div>
    );
```

Refs

The ref is used to return a reference to the element.

```
render() {
    return (
      <div>
        <input value = {this.state.data}
onChange = {this.updateState}
          ref = "myInput"></input>
        <button onClick =
{this.clearInput}>CLEAR</button>
        <h4>{this.state.data}</h4>
      </div>
    );
```

```
clearInput() {

this.setState({
data: ''});

ReactDOM.fin
dDOMNode(t
his.refs.myInp
ut).focus();
  }
```


Keys

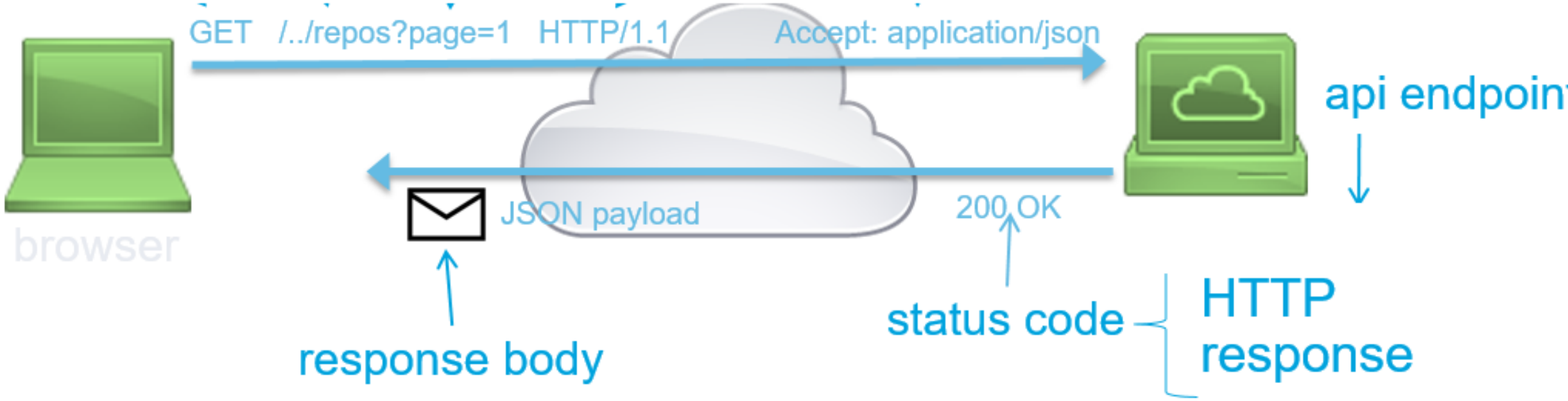
React Keys are used to dynamically create content elements with unique index(i), when the data in the list altered by the user.

```
const stringLists = [ 'Peter', 'Sachin', 'Kevin', 'Dhoni', 'Alisa'
];

const updatedLists = stringLists.map((strList, index)=>{
    <li key={index}> {strList} </li>;
});
```

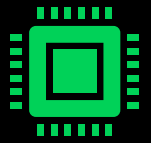

Rest API and React

Rest API and React



Rest API and React

Consuming REST APIs in a React Application can be done in various ways.



The most popular methods known as Axios (a promise-based HTTP client) and Fetch API (a browser in-built web API).

Consuming APIs Using The Fetch API

01

The fetch() API is an inbuilt JavaScript method for getting resources from a server or an API endpoint.

02

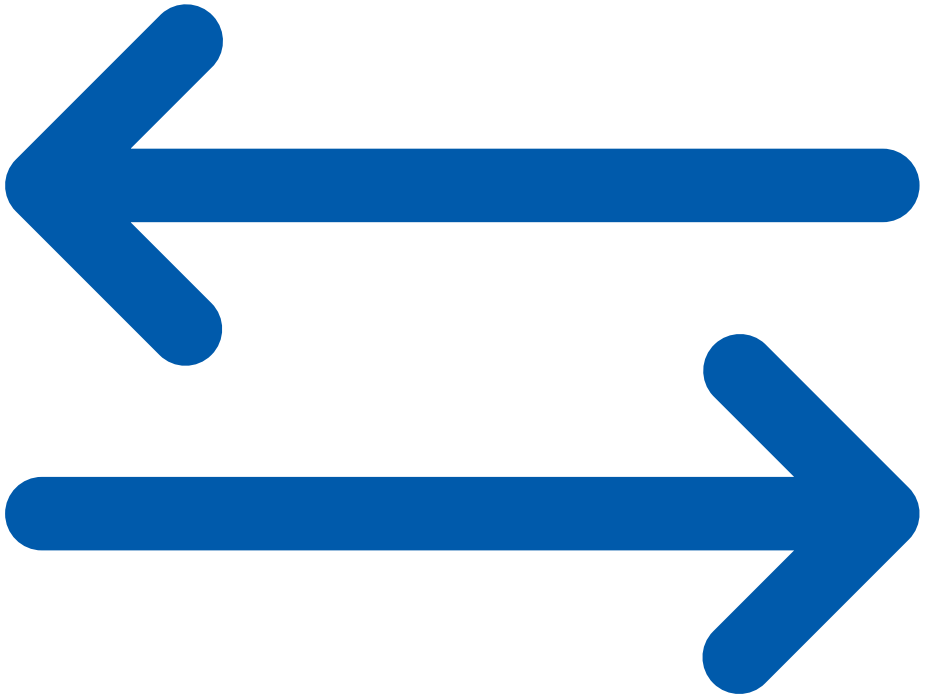
It defines concepts such as CORS and the HTTP Origin header semantics.

03

Once a response has been fetched, the inbuilt methods used to handle the data.

BASIC SYNTAX FOR USING THE FETCH() API

- fetch('http://jsonplaceholder.typicode.com/users')
- .then(response => response.json())
- .then(data => console.log(data));



PARAMETERS FOR THE FETCH API

resource
Path to the resource you want to fetch, this can either be a direct link to the resource path or a request object

init
An object containing any custom setting or credentials you'll like to provide for your fetch() request.

method
This is for specifying the HTTP request method e.g GET, POST, etc.

headers
It is for specifying any headers you would like to add to your request, usually contained in an object or an object literal.

PARAMETERS FOR THE FETCH API

body
It is for specifying a body that you want to add to your request: this can be a Blob, BufferSource, FormData, URLSearchParams, USVString, or ReadableStream object

mode
It is for specifying the mode you want to use for the request, e.g., cors, no-cors, or same-origin.

credentials
It is for specifying the request credentials you want to use for the request, this option must be provided if you consider sending cookies automatically for the current domain.

USING FETCH API IN REACT APPS

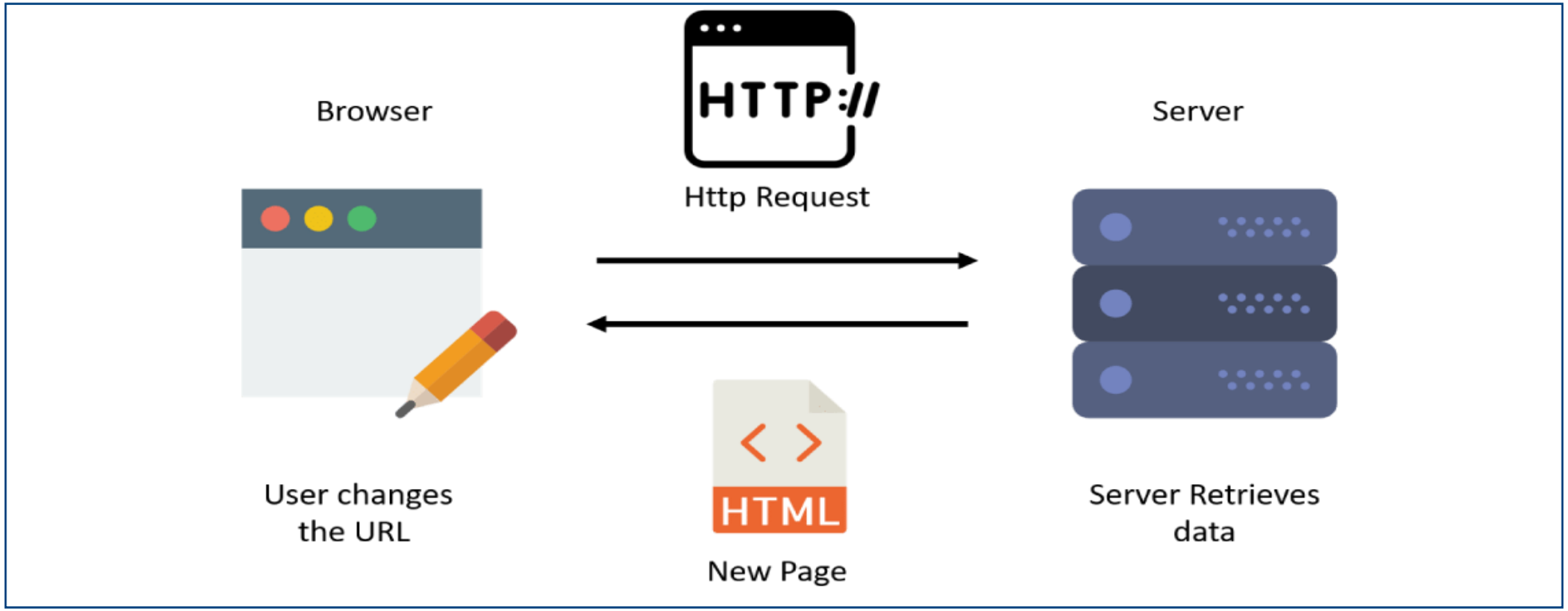
This request can either be made inside a Lifecycle Method if your component is a Class Component or inside a useEffect() React Hook if your component is a Functional Component.

The fetch request will be made inside a componentDidMount lifecycle method to make the request just after our React Component has mounted.

```
class myComponent extends React.Component {
  componentDidMount() {
    const apiUrl =
'http://jsonplaceholder.typicode.com/users';
    fetch(apiUrl)
      .then((response) => response.json())
      .then((data) => console.log('This is your data', data));
  }
```

REACT ROUTER

React Router



React Router is a standard library for routing in React. It enables the navigation among views of various components in a React Application.

Router

```
$ npm install react-router-dom --save
```

Types of router components:

<BrowserRouter>: It is used for handling the dynamic URL.

<HashRouter>: It is used for handling the static request.

Routing Steps

Step 1:

create Home component, AboutUs Component and ContactComponent along with App.js.

Step 2:

For Routing, import the components created

(e.g. Home component, AboutUs Component and ContactComponent)

import Route,Link and BrowserRouter from react-router-dom

Step-3: Open the project location in terminal type npm start.

Step 4: Add the '/about and '/contactUs' in Home Component

Link Component

The component is used to create links to navigate on different URLs and render its content without reloading the webpage.

```
const routing = (
  <Router>
    <div>
<ul>
        <li>
          <Link to="/">Home</Link>
        </li>
        <li>
          <Link to="/about">About</Link>
        </li>
        <li>
          <Link to="/contact">Contact</Link>
        </li>
      </ul>
      <Route exact path="/" component={App} />
      <Route path="/about" component={About} />
      <Route path="/contact" component={Contact} />
    </div>
  </Router>
)
```

Thank you

Innovative Services

Passionate Employees

Delighted Customers