ReactJS - Architecture

React library is built on a solid foundation. It is simple, flexible and extensible. As we learned earlier, React is a library to create user interface in a web application. React's primary purpose is to enable the developer to create user interface using pure JavaScript. Normally, every user interface library introduces a new template language (which we need to learn) to design the user interface and provides an option to write logic, either inside the template or separately.

Instead of introducing new template language, React introduces three simple concepts as given below −

React elements

JavaScript representation of HTML DOM. React provides an API, ***React.createElement*** to *create React Element*.

JSX

A JavaScript extension to design user interface. JSX is an XML based, extensible language supporting HTML syntax with little modification. JSX can be compiled to React Elements and used to create user interface.

React component

React component is the primary building block of the React application. It uses React elements and JSX to design its user interface. React component is basically a JavaScript class (extends the ***React.component*** class) or pure JavaScript function. React component has properties, state management, life cycle and event handler. React component can be able to do simple as well as advanced logic.

Let us learn more about components in the React Component chapter.

Workflow of a React application

Let us understand the workflow of a React application in this chapter by creating and analyzing a simple React application.

Open a command prompt and go to your workspace.

```
cd /go/to/your/workspace
```

Next, create a folder, *static_site* and change directory to newly created folder.

```
mkdir static_site
cd static_site
```

Example

Next, create a file, *hello.html* and write a simple React application.

```
<!DOCTYPE html>
<html>
   <head>
      <meta charset="UTF-8" />
      <title>React Application</title>
   </head>
   <body>
      <div id="react-app"></div>
      <script src="https://unpkg.com/react@17/umd/react.development.js" crossorigin></script>
      <script src="https://unpkg.com/react-dom@17/umd/react-dom.development.js" crossorigin></sc
      <script language="JavaScript">
         element = React.createElement('h1', {}, 'Hello React!')
         ReactDOM.render(element, document.getElementById('react-app'));
      </script>
   </body>
</html>
```

Next, serve the application using serve web server.

```
serve ./hello.html
```

Output

Next, open your favorite browser. Enter **http://localhost:5000** in the address bar and then press enter.

React Hello

Image not found or type unknown

Let us analyse the code and do little modification to better understand the React application.

Here, we are using two API provided by the React library.

React.createElement

Used to create React elements. It expects three parameters −

- Element tag
- Element attributes as object
- Element content - It can contain nested React element as well

ReactDOM.render

Used to render the element into the container. It expects two parameters −

- React Element OR JSX
- Root element of the webpage

Nested React element

As ***React.createElement*** allows nested React element, let us add nested element as shown below −

Example

```
<script language="JavaScript">
   element = React.createElement('div', {}, React.createElement('h1', {}, 'Hello React!'));
   ReactDOM.render(element, document.getElementById('react-app'));
</script>
```

Output

It will generate the below content −

```
<div><h1> Hello React! </h1></div>
```

Use JSX

Next, let us remove the React element entirely and introduce JSX syntax as shown below −

```
<!DOCTYPE html>
<html>
   <head>
      <meta charset="UTF-8" />
      <title>React Application</title>
   </head>
   <body>
      <div id="react-app"></div>
      <script src="https://unpkg.com/react@17/umd/react.development.js" crossorigin></script>
      <script src="https://unpkg.com/react-dom@17/umd/react-dom.development.js" crossorigin></sc
      <script src="https://unpkg.com/@babel/standalone/babel.min.js"></script>
      <script type="text/babel">
         ReactDOM.render(
            <div><h1>Hello React!</h1></div>,
            document.getElementById('react-app')
         );
     </script>
   </body>
</html>
```

Here, we have included babel to convert JSX into JavaScript and added *type=“text/babel”* in the script tag.

```
<script src="https://unpkg.com/@babel/standalone/babel.min.js"></script>
<script type="text/babel">
   ...
   ...
</script>
```

Next, run the application and open the browser. The output of the application is as follows −

Hello Jsx
Image not found or type unknown

Next, let us create a new React component, Greeting and then try to use it in the webpage.

```
<script type="text/babel">
   function Greeting() {
      return <div><h1>Hello JSX!</h1></div>
   }
   ReactDOM.render(<Greeting />, document.getElementById('react-app') );
</script>
```

The result is same and as shown below −

Hello Jsx
Image not found or type unknown

By analyzing the application, we can visualize the workflow of the React application as shown in the below diagram.

Workflow Jsx
Image not found or type unknown

React app calls ***ReactDOM.render*** method by passing the user interface created using React component (coded in either JSX or React element format) and the container to render the user interface.

ReactDOM.render processes the JSX or React element and emits Virtual DOM.

Virtual DOM will be merged and rendered into the container.

Architecture of the React Application

React library is just UI library and it does not enforce any particular pattern to write a complex application. Developers are free to choose the design pattern of their choice. React community advocates certain design pattern. One of the patterns is Flux pattern. React library also provides lot of concepts like Higher Order component, Context, Render props, Refs etc., to write better code. React Hooks is evolving concept to do state management in big projects. Let us try to understand the high level architecture of a React application.

React App
Image not found or type unknown

- React app starts with a single root component.
- Root component is build using one or more component.
- Each component can be nested with other component to any level.
- Composition is one of the core concepts of React library. So, each component is build by composing smaller components instead of inheriting one component from another component.
- Most of the components are user interface components.
- React app can include third party component for specific purpose such as routing, animation, state management, etc.

Revision #1
Created 14 December 2022 10:37:15 by Admin
Updated 14 December 2022 10:37:49 by Admin