Assignment :- 1

Q-1 What is props? How props can be passed to class and functional components?

Ans :- Props stands for " properties ". They are read-only components. it is an object which stores the value of attributes of a tag and work similar to the Html attributes. it gives a way to pass data from one to components other components. it is a similar to function arguments. props are passed to the component in the same way as arguments passed in a function.

→ Props are immutable so we cannot modify the props from inside the component.

→ inside the components, we can add attributes is called props.

→ These attributes are available in the component as this. props and can be used to render dynamic data in our render method.

→ when you need immutable data in the component, you have to add props to reactDOM. render () method in the main.js file of your React-JS project and used it inside the component in which you need.

→ React components use props to communicate with each other.

→ Every parent component can pass one information to its child components by giving them props.

→ props might remind you of , Html attributes , but you can pass any javascrip value through them , including objects, arrays and functions.

→ In order to to make props accessible i a function component you need to pass the props argument to the function and then you can acess props by using the props object inside the function.

- <u>**Passing props to Functional Components**</u> :-

```
import React from 'react';
const Greeting = (props) => {
return <h1> Hello , {props . name } ! </h1>;
};
Const APP = () => {
return < Greeting name = " Ram" / > ;
};
export default APP;
```

→ we have a functional component called ' Greeting', which receives the 'props' objects as its parameter.

- <u>**Passing props to class component**</u> :-

```
import React, { Component } from 'react';
class Greeting extends Component {
render () {
return <h1> Hello , { this. props . name }! </h1> ;
}
}
```

```
class APP extends component {
 render () {
return < Greeting name = " Ram" / >;
}
}.
```

→ In both cases the output will be same as :-

Hello, Ram !

Q-2 Write Detail note on MVC architecture.

Ans :- MVC (Model - view - controller) is a software architectural pattern that divides an application into three interconnected components to specific concerns and promote, a modular and organized design. it is widely used in the development of web and desktop application.

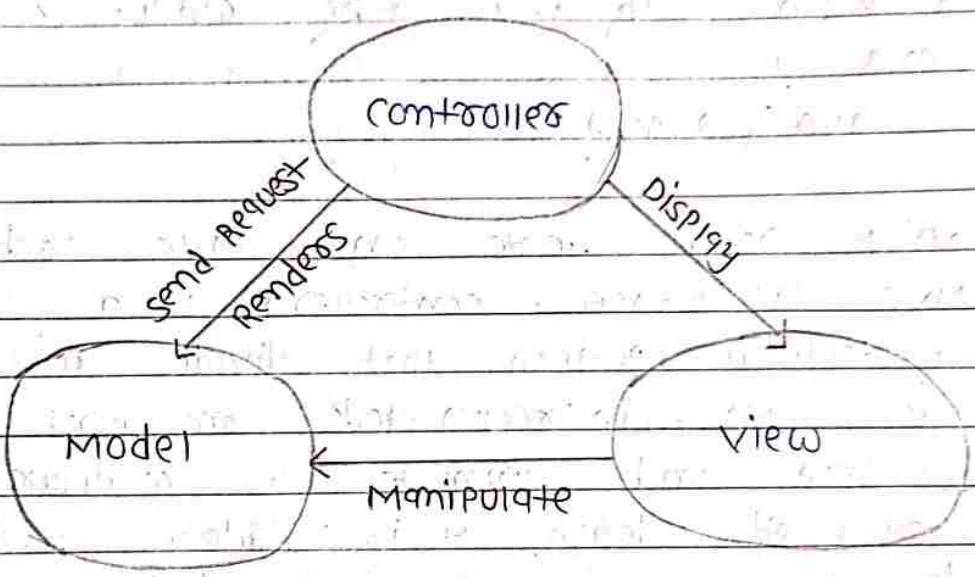
- MVC Architecture :-

1. Model :-

The Model represents the application's data and business logic. it encapsulates the data and business methods to interact with the manipulate that data. This component is responsible for managing the application's state, data validation and database operations.

2. View :-

The view represents the user interface of the application. it displays the data from the model to the user and captures user input, but it should not contain any business logic The primary purpose of the view is to present to the information users in a visualy appealing and understandable way.



[MVC - Architecture]

3. Controller :-

The controller acts as an intermediary between the model and the view. it receives user input from the view and processes it, updating the model accordingly. The controller contains the application's business logic, handling user request, coordinating with the model to retrive or update data and deciding which view to render.

Q-3 Differentiate between :-

• Real DOM and virtual DOM

Real DOM	virtual DOM
(1) DOM manipulation is very expensive.	(1) DOM manipulation is very ease.
(2) There is too much memory wastage	(2) NO memory wastage.
(3). it updates slow	(3). it updates fast
(4). it can directly update HTML	(4). it can't update HTML directly.
(5). creates a new DOM if the element updates	(5) update the JSX if the element update
(6). it allow us to directly target any specific node.	(6). it can produce about 200,000 virtual DOM nodes
(7). it represents the UI of your application	(7). it is the virtual DOM representation of the DOM.

• useMemo and usecallback hooks

useMemo	usecallback
(1) Memoizes the result of a function call	(1) Memoizes a function
(2) Returns the memoized value	(2). Returns the memoized function
(3). used when you want to avoid re-computing a value on every render.	(3). used when you want to prevent unnecessary re-creation of a function.
(4) Dependencies are provided to control when the value should be recalculated.	(4). Dependencies are provided to control when the function should be re-created.
(5). useMemo focuses on avoiding heavy calculation.	(5). usecallback focuses on avoiding different thing.

Assignment :- 2

Q-1 Write note on :-

- **Angular Filters And Controller :-**

1. **Angular Filters :-**

In AngularJS, filters are used to format data. Following is a list of filters used for transforming data.

(1). Currency :- it formats a number to a currency format.
(2). Date :- it formats a date to a specified format.
(3). Filter :- it select a subset of item from an array.
(4). JSON :- it formats an object to a JSON string.
(5). Limit :- it is used to limit an array into a specified number of element.
(6). Lowercase :- it formats a string to lower case.
(7). Number :- it formats a number to a string.
(8). orderby :- it orders an array by an expression.
(9). Uppercase :- it formats a string to uppercase.

2. **Controller :-**

Angular JS controllers control the data of Angular JS applications. Angular JS controller are regular Javascript objects.

AngulaR JS applications are controlled by controllers. The ng- controller directive defines the application controller. A controller is a Javascript object created by a standard Java script object constructor.

→ Each controller accepts $ scope as a which parameter refers to the application / module that controller is to control.

• **Example of controller :-**

```
< div ng- app = " myAPP" ng - controller= "myctrl">
First Name :-
< input type = "text" ng- model = "firstName">
<br>
Last Name :-
<input type= "text" ng- model = " lastName">
< br >
Full Name :-
{{ firstName + " " + lastName }}
</ div >
< script >
var app = angular . module ('myAPP', [] );
app. controller ('myctrl', function ($scope) {
$ scope .firstName = " Dhiru";
$ scope . lastName = " saii ";
});
</script >
```

→ Here is the example of controller of AngulaR. JS.

→ In above example ng- controller is used.

• **Hooks with Advantages :-**

React Hooks are a type of function that allows the users to hook into react state and life-cycle features from function components. React provides built-in hooks like usestate, useeffect, useReducer, useRef, usecallback, usecontext, and useMemo and we can also create your own custom hooks.

⟶ React hooks are available from React version of 16.8.

⟶ Before the introduction of hooks, the state can be maintained only in the class component not in the functional component

⟶ After the introduction of hooks, the state can be maintained in functional component too.

⟶ The idea of using hooks makes it possible to create full-fledged functional component while using all React features.

• **Advantages of Hooks :-**

1. **Modularity :-**

Hooks promote modularity by enabling developers to separate different concerns and functionalities into discreate units. Helps organize code into separate units.

2. Reusability :-

Enables sharing of code across different parts of the application. This reduces redundent code and prompts a more efficent development process.

3. Customization :-

Easily adapt the software to specific requironment. Hooks allow for easy customization and extensibility of an application.

4. Testing :-

Facilities easier testing and debugging. this can improve the overall quality and reliability of the application.

5. Separation of concerns :-

Hooks support the principle of separation of concerns, where different parts of an application handle distinct functions.

6. Readability :-

Hooks can improve code readability by encapsulating specific functionalities within named hooks.

7. Performance :-

well - designed hooks can lead

to improved performance since they can be optimized independently. Allows optimization of critical functionalities independently.

8. <u>Community contributions</u> :-

HOOKS encourge a modular and extensible design, making it easier for the community to contribute to open - source projects.

Q-2 Explain React components in detail.

<u>Ans</u> :- components are like functions that return HTML elements. components are independent and reusable bits of code. They serve the same purpose as Javascript functions, but work in isolation and return HTML.

→ component is considered as the core building blocks of a React application.

→ it makes the task of building UIs much easier.

→ Each component exists in the same space, but they work independently from one another and merge all in a parent component, which will be the final UI of your application

→ Every React component have their own structure, methods as well as APIS.

→ They can be reusable as per your need, for better understanding, consider the entire UI as a tree.

→ Here, the root is the starting component and each of the other pieces becomes branches, which are further divided into sub-branches.

In React JS, we have mainly two types of components They Are.

1. Functional components
2. Class components.

1. Functional Components :-

In React, function components are a way to write components that only contain a render method and don't have their own state. They are simply javascript functions that may or may not receive data as parameters. we can create a function that takes props as input and returns what should be rendered.

• Example :-

```
function welcomeMessage (props) {
return <h1> Hello, {props. name} </h1>;
}
```

→ The functional component is also known as a stateless component because they do not hold or manage state.

2. Class Components :-

A class component must include the extends React. component statement. This statement creates an inheritance to React component, and gives your component access to React. component's functions. The component also require a render () method, this methods returns Html. class component are more complex than functional components.

- Example :-

```
class car extends React. component {
render () {
return <h1> Hi, I am a car! </h1>;
}
}
```

⟶ Here above example CAR is extends by React. component in the class component

Q-3 What is an angular directive ? Explain ng-app, ng-model and ng-bind with example.

Ans :-

Angular JS facilities you to extend html with new attributes. These attributes are called directives. There is a set of built-in directive in Angular JS which offers functionality to your applications. You can also define your own directive. Directive are special attributes starting with ng-prefix.

→ Angular directive begin with ng- where ng stands for Angular and extends html tags with @directive decorator.

→ In Angular Directives are defined as classes that can add new behavior to the elements in the template or modify existing behavior.

→ The purpose of directives in Angular is to maneuver the DOM, be it by adding new elements to DOM or removing elements and even changing the appearance of the DOM elements.

- **ng - app** :-

The ng - app directive is used to define the root element of an Angular JS application. it specifies the name of the Angular JS module to be used in the application.

Example :-

```
<html ng-app = "myAPP">
<head>
<script src = "https :// ajax . com "> </script>
</head>
<body>
<div ng- controller = "myCtrl">
<p> {{ greeting}} </p>
</div>
<script>
```

```
var app = angular.module ('myApp', []);
app.controller ('myCtrl', function ($scope) {
$ scope . greeting = " Hello world";
});
</script>
</body>
</html>
```

→ in this example, the ng-app directive is used to define the root element of the Angular JS application.

• **ng - model** :-

The ng-model directive is used to bind the value of an HTML control (input, select, textarea) to a variable defined in an Angular JS controller. The ng-model directive is added to an HTML control such as an input field.

→ The value of the ng-model directive is the name of the variable that will be used to store the value of the HTML control.

Example :-

```
< html ng-app >
< head>
< script src = " https :// ajax . com / 1.6.9 angular.
  min.js "> </ script>
</head>
< body>
<div ng-controller = "myCtrl">
```

```
<input type = "text" ng-model = "name">
<p> Hello, {{name}}! </p>
</div>
<script>
angular . module ('myApp', []). controller
('myCtrl', function ($scope) {
 $ scope . name = "world";
});
</script>
</body>
</html>
```

→ In this example, the ng-model directive is used to bind the value of the input field to the $ scope . name variable.

- **ng - bind** :-

The ng-bind directive is used to bind the content of an HTML element to a variable in Angular js. The ng-bind directive is added to an HTML element, such as a paragraph or a div. The value of the ng-bind directive is the name of the variable that will be used to populate the content of the HTML element.

Example :-

```
<html ng-app >
<head>
< script src = " https :// ajax . google. com /
  1.6.9 / angular . min.js "> </ script>
```

```
</head>
<body>
<div ng- controller = "myCtrl " >
<p ng-bind = " greeting "> </p>
</div>
<script>
angular . module ('myAPP', []) . controller
('myCtrl', function ($sope) {
  $ scope . greeting = " Hello, world! ";
});
</script>
</body>
</html>
```

→ In this example, the ng-bind directive is used to bind the content of the paragraph element to the $ Scope . greeting variable.

→ The value of the variable is initially set to " Hello, world! " in the controller.

→ when the page loads, Angular JS evaluates the expression in the ng-bind directive and popules the content of the paragraph element with the value of the variable.