**UNIT**-V

* Discuss about Installation of Node.js and working with Node Packages
* Describe Angular, Modules, Directives and Data Binding
* Illustrate Dependency Injection, Services, and Creating a Basic Angular Application.

**Installation of Node.js and working with Node Packages**

Node.js is a JavaScript runtime built on Chrome's V8 JavaScript engine. It allows you to execute JavaScript code on the server-side.

Node.js comes with a package manager called npm (Node Package Manager) that simplifies the installation and management of third-party libraries and tools.

**Installation Steps:**

**Download Node.js:**

Visit the official Node.js website (https://nodejs.org/).

Download the LTS (Long Term Support) version, which is recommended for most users.

**Install Node.js:**

Run the downloaded installer package.

Follow the installation instructions in the installer wizard.

Node.js will be installed along with npm.

**Verify Installation:**

Open your command prompt or terminal.

Run the following commands to check the installed versions:

node -v // Check Node.js version

npm -v // Check npm version

**Updating npm (optional):**

You can update npm to the latest version using the following command:

npm install -g npm

**Working with Node Packages (npm)**

npm Basics:

npm is a package manager for Node.js, used for installing and managing JavaScript packages and libraries.

**Common npm Commands:**

1. **Installing Packages:**

To install a package locally in your project, use:

npm install package-name

To install a package globally (accessible from any project), use the -g flag:

npm install -g package-name

**2.Package.json:**

* The **package.json** file is a configuration file for your project that lists dependencies and project metadata.
* You can create one using:

npm init

To install dependencies listed in **package.json**, use:

npm install

**3.Managing Dependencies:**

* Installed packages and their versions are listed in **package.json**.
* Use **--save** or **-S** to save a package as a dependency:

npm install package-name --save

Use **--save-dev** or **-D** to save a package as a development dependency (used during development but not in production):

npm install package-name --save-dev

**4.Uninstalling Packages:**

To remove a package, use:

npm uninstall package-name

**5.Viewing Package Information:**

* To view information about a package, use:

npm info package-name

**6.Updating Packages:**

* To update packages, use:

npm update

**Create Your Node.js Application:**

* Create a directory for your Node.js application and navigate to it using your terminal or command prompt.
* Initialize a Node.js Project: If your project is new, you can initialize it with npm by running the following command in your project directory:

**npm init**

* This command will prompt you to answer a series of questions to create a package.json file, which will contain information about your project and its dependencies.
* Create Your Application Code: Write your Node.js application code in a JavaScript file, e.g., app.js.
* Install Dependencies: If your application depends on external libraries or packages, you can install them using npm. For example, to install the Express.js framework, you can run:

**npm install express**

* This will add Express.js as a dependency in your package.json file.
* Write Your Application Code: Write your Node.js application code in the app.js file. Here's a simple example of a Node.js application using Express.js:

const express = require('express');

const app = express();

const port = 3000;

app.get('/', (req, res) => {

res.send('Hello, World!');

});

app.listen(port, () => {

console.log(`Server is listening at http://localhost:${port}`);

});

* Run Your Application: To run your Node.js application, open your terminal or command prompt, navigate to your project directory, and then run the main JavaScript file using Node.js. In this example, you would run:

**node app.js**

* Your server will start, and you can access it by opening a web browser and navigating to http://localhost:3000 (assuming you used port 3000 in your code). You should see "Hello, World!" displayed in your browser.
* Manage Your Application: While your application is running, you can make changes to your code, and the changes will take effect once you restart the Node.js process. You can stop the server by pressing Ctrl + C in the terminal.

**AngularJS**

* AngularJS is a JavaScript framework. It can be added to an HTML page with a <script> tag.
* AngularJS extends HTML attributes with Directives, and binds data to HTML with Expressions.
* AngularJS is a JavaScript framework written in JavaScript.
* AngularJS is distributed as a JavaScript file, and can be added to a web page with a script tag:

**<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></script>**

**AngularJS Extends HTML**

* AngularJS extends HTML with ng-directives.
* The ng-app directive defines an AngularJS application.
* The ng-model directive binds the value of HTML controls (input, select, textarea) to application data.
* The ng-bind directive binds application data to the HTML view.

**AngularJS Example**

<!DOCTYPE html>

<html>

<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></script>

<body>

<div ng-app="">

<p>Name: <input type="text" ng-model="name"></p>

<p ng-bind="name"></p>

</div>

</body>

</html>

* AngularJS starts automatically when the web page has loaded.
* The ng-app directive tells AngularJS that the <div> element is the "owner" of an AngularJS application.
* The ng-model directive binds the value of the input field to the application variable name.
* The ng-bind directive binds the content of the <p> element to the application variable name.