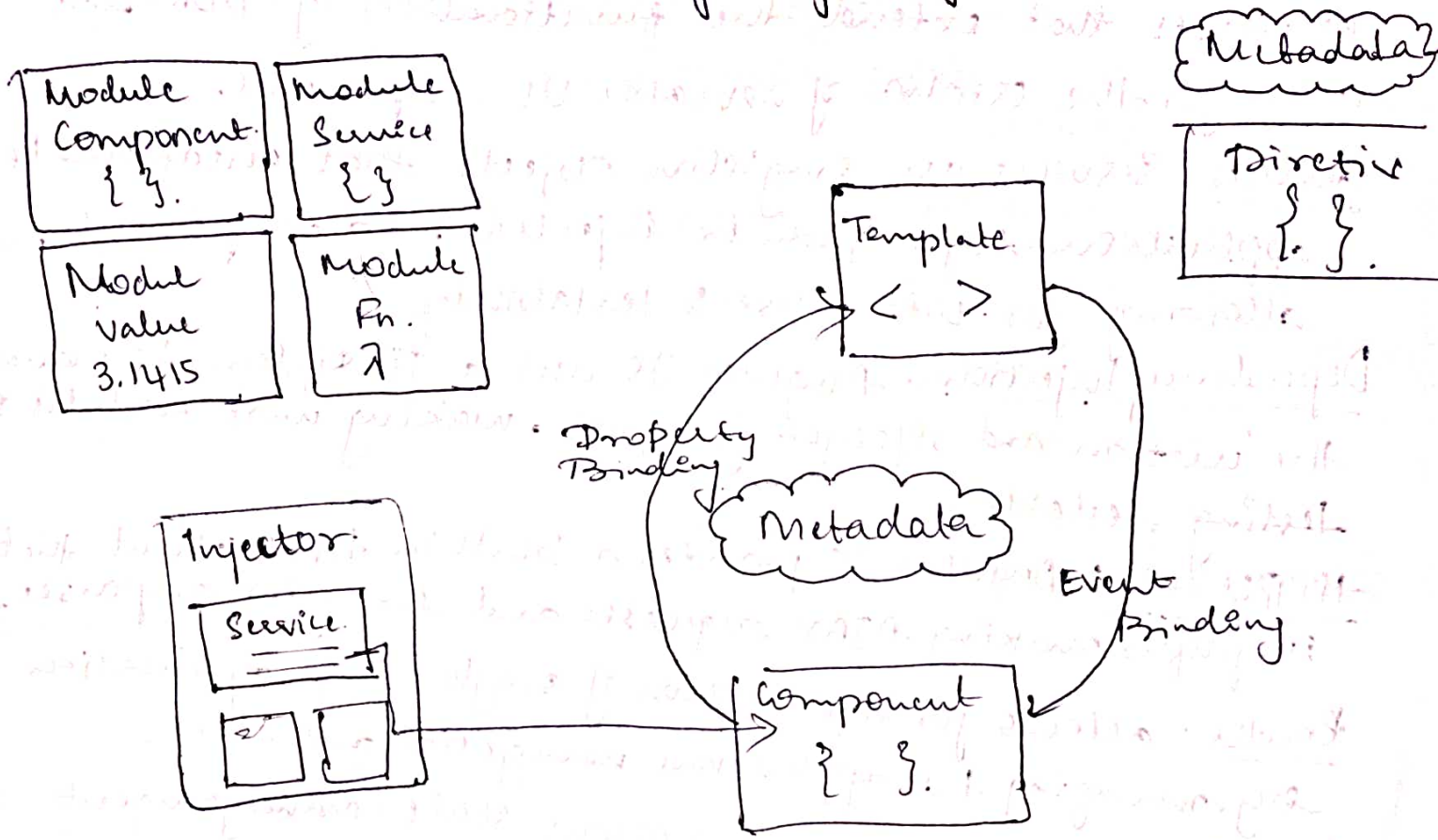
1. Explain the architecture of Angular.js?



AngularJS follows the Model-View-Controller (MVC) architectural pattern to organize the application's structure and functionality.

Modules: AngularJS applications are divided into smaller, reusable modules that encapsulate specific functionality. This allows for better organization and modularity.

Components: bring together the view, logic, and data into a reusable building block for the application.

Templates: define the HTML structure and layout of the application's view. They use directives, expressions, and data binding to make the views dynamic.

Directives: Directives are custom HTML elements, attributes or classes that extend the functionality of DOM and allow for the creation of reusable UI components.

Services: Services are singleton objects that encapsulate application logic & can be injected into components. allowing for code reuse & testability.

Dependency Injection: Angular JS uses a DI system to manage the creation and lifecycle of objects., making more modular & ~~testing~~ testable.

HTTPClient: Angular JS provides a built in HTTP client that simplifies making AJAX requests and handling responses.

Routing: allows for the creation of single-page application by managing the applications navigation and state.

State Management: can use various state management techniques such as two-way data binding to keep the UI in sync with the application's state.

2) How does an Angular application work?

In Angular application, a module is nothing but a collection of components, directives, pipes, and services that are typically used in the application.

modules: group components, directives, pipes and services. Root module ("AppModule") bootstraps the application.

components: control parts of the UI

Root component connects component hierarchy with the DOM.

Services : Provide reusable data or logic.
Use the '@Injectable' decorator.

Templates : combine HTML and Angular markup
Modify HTML elements and connect data to the DOM.

Metadata : Informs Angular how to process a class.
Decorates classes to configure their behavior.

Data Binding : Facilitates real-time communication b/w templates and components. Allows communication b/w parent and child components.

Directives : Add behavior to elements in templates. Essential for building dynamic UI components.

3) What are directives in Angular?

Directives in AngularJS are special markers (attributes, elements, comments or classes) that tell the angularJS compiler to attach a specified behavior to the DOM element (or transform it and its children). They are used to extend HTML functionality & make it more dynamic.

Type : 1. Attribute Directives.
2. Structural Directives
3. Class Directives
4 Element Directives.

4). Explain components, Modules and Services in Angular?

Components : Components are the fundamental building blocks of an angular application. They control a part of the UI.

Key Elements : * Selector : Defines how the component is used in HTML.
* Template : The HTML view controlled by the component.
* Class : contains the business logic for the component.
* Styles : CSS styles specific to the component.

Modules : are containers for a cohesive block of code dedicated to an application domain, a workflow / set of closely related capabilities.

Key Elements : * NgModule Decorator : Marks a class as an Angular module & provides configuration metadata
- * Imports : List of other modules whose exported classes are needed by components in this module.
- * Providers : List of services used in this module.
- * Bootstrap : Main application view called the root component, which host all other app views.

Services : are classes that handle the business logic and are used to share data & functionality across components.

Key Elements : @Injectable Decorator : marks a class as available to be provided & injected as a dependency.

* Methods & Properties : Define the functionality and data that the service provides.

5) What are some advantages of Angular over other frameworks ?

* Two way Data Binding - Automatically synchronizes data b/w the model & the view.

* Dependency Injection : makes the code more modular, maintainable and testable by managing service dependencies.

* Directives : Extends HTML with custom attributes & elements for dynamic content.

* MVW Pattern (Model-View-whatever) supports various design patterns like MVC / MVVM making it flexible.

* Built-in Services : Provides various built-in services that simplify common ~~mistake~~ tasks.

* Ease of Testing
Designed with testing in mind, support both unit testing & end-to-end testing.