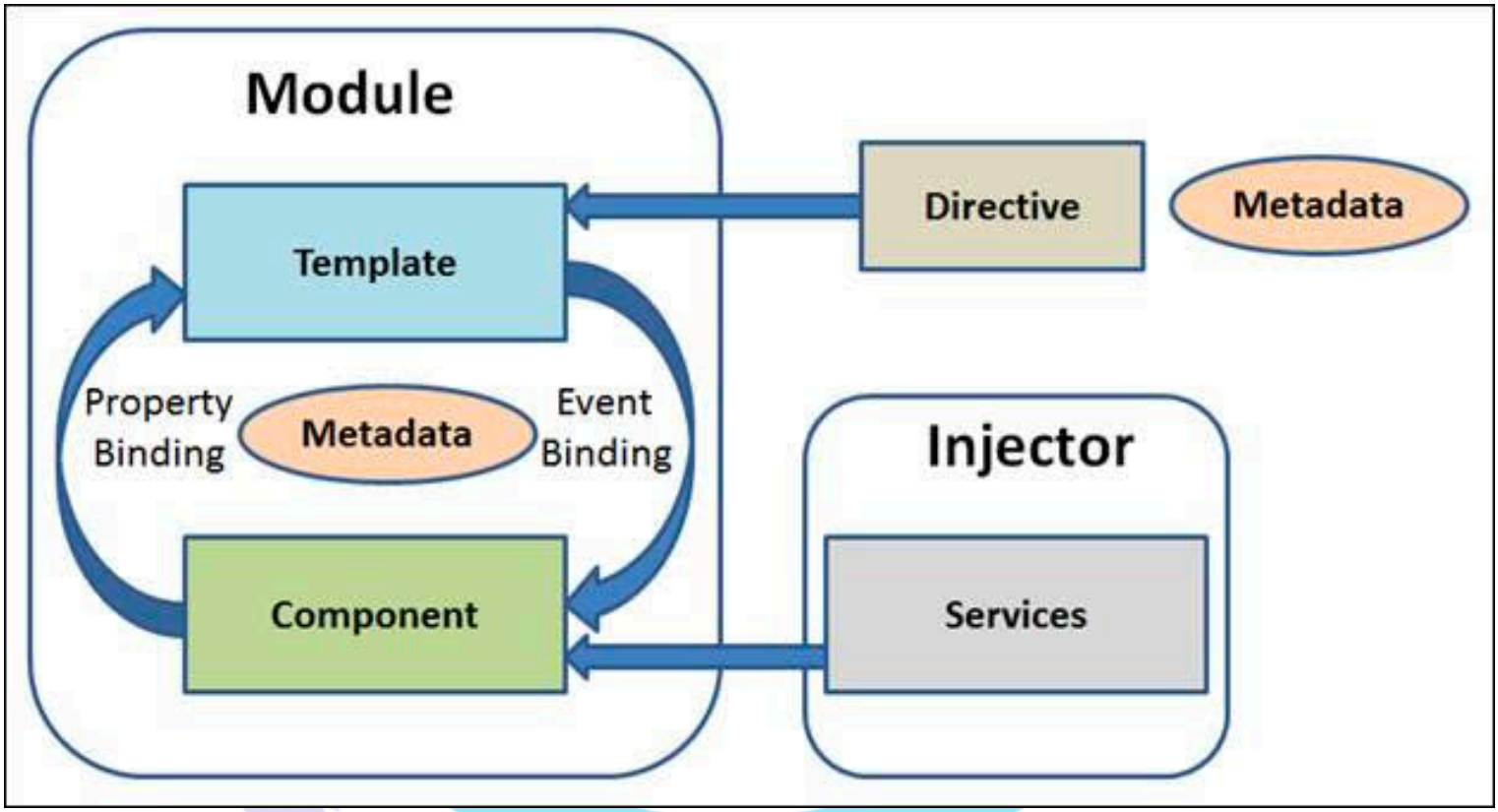
Angular Service

Service is a broad category encompassing any value, function, or feature that an app needs. A service is typically a class with a narrow, well-defined purpose. It should do something specific and do it well.

Angular distinguishes components from services in order to increase modularity and reusability.



• By separating a component's view-related functionality from other kinds of processing, you can make your component classes lean and efficient. Ideally, a component's job is to enable the user experience and nothing more. It should present properties and methods for data binding, in order to mediate between the view (rendered by the template) and the application logic (which often includes some notion of a model).

• A component should not need to define things like how to fetch data from the server, validate user input, or log directly to the console. Instead, it can delegate such tasks to services. By defining that kind of processing task in an injectable service class, you make it available to any component. You can also make your app more adaptable by injecting different providers of the same kind of service, as appropriate in different circumstances.

Angular doesn't enforce these principles. Angular does help you follow these principles by making it easy to factor your application logic into services and make those services available to components through dependency injection.

Angular Dependency Injection

Components consume services; that is, you can inject a service into a component, giving the component access to that service class.

To define a class as a service in Angular, use the @Injectable decorator to provide the metadata that allows Angular to inject it into a component as a dependency.
Similarly, use the @Injectable decorator to indicate that a component or other class (such as another service, a pipe, or an NgModule) has a dependency. A dependency doesn't have to be a service—it could be a function, for example, or a value.

Dependency injection (often called DI) is wired into the Angular framework and used everywhere to provide new components with the services or other things they need.

• The injector is the main mechanism. You don't have to create an Angular injector. Angular creates an application-wide injector for you during the bootstrap process.

• The injector maintains a container of dependency instances that it has already created, and reuses them if possible.

• A provider is a recipe for creating a dependency. For a service, this is typically the service class itself. For any dependency you need in your app, you must register a provider with the app's injector, so that the injector can use it to create new instances.