Difference Between Angular Components and Services.

|  |  |  |
| --- | --- | --- |
| **Points** | **Angular Components** | **Angular Services** |
| **Definition and Role** | Components are the fundamental building blocks of an Angular application.  They control a portion of the screen (a view) through their associated template. | Services are broad-scope classes designed to provide specific functionality unrelated to views and UI manipulation.  They can be injected into components and other services through Angular's dependency injection system. |
| **Structure** | A component is defined using the ‘@Component’ decorator which includes properties that specify the associated HTML template, styling, and more.  Typically includes a TypeScript class, an HTML template, and often accompanying CSS styles. | A service is typically defined using the ‘@Injectable’ decorator, indicating that it can be injected into other classes.  Consists primarily of a TypeScript class with methods that perform specific functions. |
| **Responsibilities** | Components are responsible for interaction with the template (view) to render application data and handle user interactions.  They manage data binding, and event binding, and directly interact with the view and user inputs. | Services handle data management (fetching data, manipulating data), logging, analytics, and other business logic.  They are designed to be independent of any particular view, and their methods can be reused across different components. |
| **Lifecycle** | Angular manages the lifecycle of components through specific lifecycle hooks (e.g., ‘ngOnInit’, ‘ngOnDestroy’), allowing developers to perform actions at different stages of the component's lifecycle. | Unlike components, services do not have a lifecycle managed by Angular; they are instantiated as needed and exist typically for the life of the application (singleton scope usually). |
| **Scope** | Components are usually not shared across different application parts unless specifically designed to be reusable (e.g., a UI button component). | Services are designed to be reusable throughout an application. They provide a mechanism to share data and functionality across components. |
| **Data Handling** | Components handle and display data to users; they interact with services to fetch and manipulate data but ideally do not handle data persistence or extensive business logic themselves. | Services may handle state management, interact with databases or backends, and provide data to components. They serve as the intermediary between the backend and the frontend. |

Version vise difference between angular components and service.

1. **Angular 2**

* **Introduction of Components and Services**:
  + Angular 2 introduced a completely new framework compared to AngularJS (Angular 1.x). Components in Angular 2 are like building blocks for your web page; they manage what you see on the screen (the UI).
  + Services, on the other hand, are used for sharing code across components, like fetching data or logging information. Both use TypeScript, which helps with providing additional safety and tools for developers.

1. **Angular 6 to Angular 9**

* **Improvements to Services**:
  + Angular 6 introduced something called "Tree Shakable Providers". This was a fancy way of making services optional—if you don't use a service, it doesn't get included in your final website, which helps make your website load faster.
* **Better Performance and Lazy Loading**:
  + Over these versions, Angular didn't change how you write components and services but made them work better and faster. For example, you could start loading parts of your website in the background (lazy loading), making the initial load faster.

1. **Angular 10 to Angular 12**

* **Stricter Checking**:
  + Angular started to be stricter about mistakes in your code, especially in how data is handled in templates connected to components. This means fewer bugs and smoother running applications.
* **Easier Services**:
  + Angular made it easier and clearer how to provide services so that they are only included when needed. This also helps in making your application smaller and faster to load.

1. **Angular 13 to Latest**

* **Ivy Compiler**:
  + This is a new system introduced from Angular 9 that became the standard by Angular 13. It doesn't change how you write your components or services but does make them faster and smaller when turning your code into a website.
* **Enhanced Tools and Debugging**:
  + Newer versions have better tools for finding and fixing bugs, and for ensuring your code is as efficient as possible. This helps you make sure that your components and services are working well together.

1. **Angular 17**
   * Angular 17 introduces deferrable views, which allows you to load and render views only when they are needed. This can improve the performance of your application by reducing the number of views that need to be rendered at any given time.

* Angular 17 introduces standalone components, which are components that can be used without being part of an NgModule. This makes it easier to share components between applications and libraries.
* Angular 17 introduces signal, which is a new way to communicate between components and services. Signals are more efficient than observables and can be used to implement a variety of patterns, such as state management and event handling.

**Key Points Across All Versions**

* **Components**: Always focused on managing the user interface, like buttons, text, and images on the screen.
* **Services**: Consistently used to handle tasks that aren't directly tied to what the user sees, like fetching data from a server, and are shared across different parts of the application.