Angular Theory Questions

1. **What is PUT , PATH and POST**

1. PUT (Create or Replace):
Purpose: The PUT method is used to create a new resource or **replace an existing resource** with a new representation.
Idempotent: PUT is considered idempotent, meaning that making multiple identical PUT requests should have the same effect as a single request.
Usage: When using PUT, the entire representation of the resource is sent in the request payload. If the resource already exists, it is replaced entirely with the new representation sent in the request.
Idempotent (sending the same request multiple times results in the same outcome).

2. POST (Create or Append):
Purpose: The POST method is used to submit data to be processed by the resource identified by the URI. It can be used to create a new resource or **append data to an existing resource.**
Non-Idempotent: POST is considered non-idempotent, meaning that making the same POST request multiple times may result in different outcomes or create duplicate resources.
Usage: When using POST, the request payload contains the data to be processed or appended. The server determines the location of the new resource and returns the appropriate response, which may include the URI of the newly created resource.
Non-idempotent (sending the same request multiple times may create multiple resources).

3. PATCH (Update):
Purpose: The PATCH method is used to partially update an existing resource. It applies modifications to the resource, rather than replacing it entirely.
Idempotent (in theory): PATCH is intended to be idempotent, but its actual implementation can vary. In practice, it's recommended to implement PATCH requests in an idempotent manner to ensure consistent behavior.
Usage: When using PATCH, the request payload contains instructions on how the resource should be modified. The server applies these changes to the existing resource while leaving the unmodified parts unchanged.
In summary, PUT is used for creating or replacing resources, POST is used for creating or appending data to resources, and PATCH is used for partially updating existing resources.
more@Medium

2. **What is Angular Polyfils**

In the context of Angular, polyfills serve a similar purpose. Angular polyfills are **JavaScript scripts or module**s that are included in an **Angular application to ensure compatibility with older browsers.** These polyfills enable Angular applications to run smoothly and consistently across different environments, ensuring a consistent user experience.

Angular Theory Questions

more@Medium

3. **What Is Zone.js?**

   Zone.js: The Heart of Angular Polyfills

Zone.js is a critical part of **Angular's change detection mechanism**. It is a JavaScript library that provides execution context and hooks into asynchronous operations. In simpler terms, Zone.js helps Angular keep track of all the asynchronous tasks and events that occur in your application and manage their execution.

Why Is Zone.js Important?
Zone.js plays a vital role in Angular's core functionality. Here's why it's so crucial:

1. **Change Detection:** Angular uses a mechanism called change detection to update the view whenever the application's state changes. Zone.js is responsible for triggering change detection when asynchronous operations like HTTP requests, timers, or user interactions occur. Without Zone.js, you'd need to manually trigger change detection after each asynchronous task, making your codebase more error-prone and complex.
2. **Error Handling:** Zone.js helps in capturing and handling errors that occur during asynchronous operations. It provides a central place to intercept and manage errors, making debugging and error reporting more efficient.
3. **Async Tracking:** Zone.js allows Angular to track asynchronous tasks, such as promises and observables, so that it knows when these tasks complete. This is essential for ensuring that your application's UI stays in sync with the underlying data.
4. **Cross-Browser Compatibility:** Zone.js takes care of the differences in how various browsers handle asynchronous operations. It provides a consistent and reliable way to handle asynchronous code, ensuring that your Angular application behaves the same way across different browsers.

   more@Medium

4. **How change detection happened on Component -**

   zones.js or host listener

5. **How to register module in Angular ?**

   Step 1 - In **AppModule**(Main) we import a **AppRoutingModule**
   Step 2 - and in **AppRoutingModule** we route the required module according to routing

Angular Theory Questions

6. **What is AOT and JIT - which one by default support**
   - Ahead of Time/Just in Time
   - Compiled @build time / At compiled @application execution
   - faster/slower and small delay in the application to start

7. What is Angular Encapsulation ?
   >> How HTML and logic you handle together (Emulative way, None)

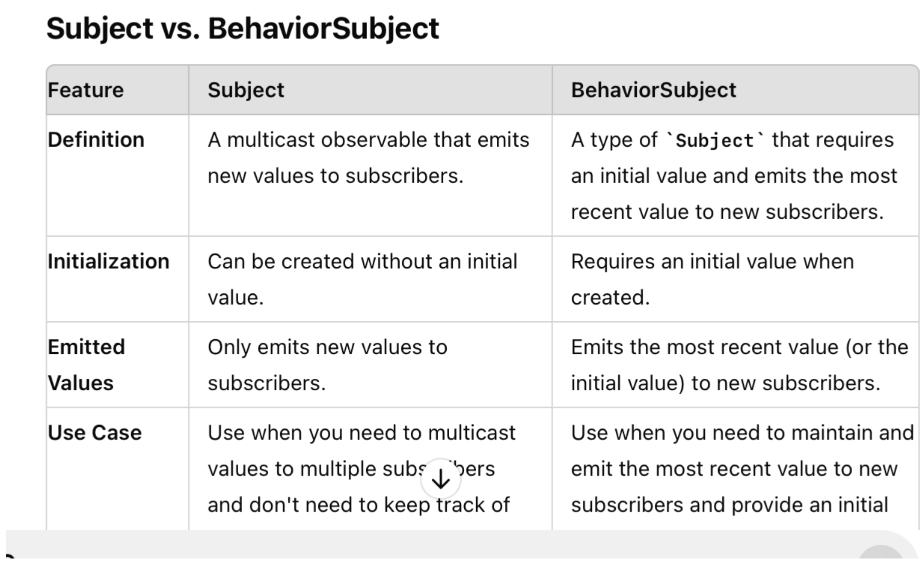
8. Lazy loading example ?
   >> explain Containerization module

9. How to handle multiple http request ?
   >> BackendToFrontend technology

10. Local Storage and Session Storage

Local Storage	Session Storage
As it is not session-based, it must be deleted via javascript or manually	It's session-based and works per window or tab. This means that data is stored only for the duration of a session, i.e., until the browser (or tab) is closed

11.



Subject vs. BehaviorSubject

Feature	Subject	BehaviorSubject
Definition	A multicast observable that emits new values to subscribers.	A type of `Subject` that requires an initial value and emits the most recent value to new subscribers.
Initialization	Can be created without an initial value.	Requires an initial value when created.
Emitted Values	Only emits new values to subscribers.	Emits the most recent value (or the initial value) to new subscribers.
Use Case	Use when you need to multicast values to multiple subscribers and don't need to keep track of	Use when you need to maintain and emit the most recent value to new subscribers and provide an initial

Angular Theory Questions

Parent to Child -

```
Parent ->  @Input() categoryType : string | undefined;
 Child -> <app-kisan-services [categoryType]="category"></app-kisan-
services>
```

Child to Parent -

```
Child  -> @Output() showCategory = new EventEmitter<string>();
Parent -> <app-filters (showCategory)="onShowCategory($event)"></app-
filters>
```

Content Sharing-
By using ngModel

11. What is wild card routes in Angular

Wildcard routes use the ** path to match any URL that hasn't been matched by preceding routes.

In Angular, the **angular.json** file is a critical configuration file used to define and manage the settings and options for your Angular projects. It contains configurations for different aspects of your Angular application, including build settings, development server options, and project structure.