## 🚀 Angular & JavaScript Interview Experience in PWC

Today, I faced a thorough interview on Angular, JavaScript, and front-end concepts. Here's a recap of what I encountered:

**Angular Topics:**

NgRx & RxJS: Observables, Subjects, debounce, throttle, streams, canceling API calls.
Authentication & Authorization: Role-based access, guards, token handling.
Interceptors: Adding headers, catching errors, modifying requests.
Memory Leaks: Avoiding closures holding unnecessary references.
Lazy Loading & Standalone Components: Route-based vs component-based lazy loading, pros/cons of standalone components.
Directives & HostListener: Creating custom directives, handling DOM events.
Signals (Angular 16): Reactive component state, sharing signals across components.
Angular Architecture: Components, modules, services, change detection.
PrimeNG vs Material UI: Differences in UI component libraries.

**JavaScript Topics:**

Closure & Lexical Scope: Preserving variables, avoiding memory leaks.
Hoisting: Behavior of var, let, const, and function declarations.
Arrow Functions: this context, naming, callbacks.
Prototype & Inheritance: Adding methods, prototype chain vs class-based inheritance.
Promises & Async/Await: Execution order, microtasks vs macrotasks, error handling.
Callbacks vs Promises: Priority in event loop, nested async examples.
Timers: setTimeout, setInterval, clearInterval.
LocalStorage & SessionStorage: Differences and use cases.

**Coding Questions I Encountered:**

RxJS Examples: Debounce input, combine multiple API calls, cancel previous request.
Signals: Counter, toggle, computed signals, arrays & objects, passing signals between components.
Prototypes: Add custom method to Array or Object.
Memory Leak: Example with closures and how to avoid them.
Async JS: Nested Promises, async/await, setTimeout, queueMicrotask—predict execution order.
Angular Interceptor: Modify request, handle errors, and chain multiple interceptors.
Lazy Loading: Implement a module-based lazy loading route.
Custom Directive: Example to change element style on hover.

Standalone Component: Create self-contained component without NgModule.
Authentication & Role-based Access: Using guards and signals/services for user roles.

**Testing & Tools:**

Jasmine & Karma: Spies, spyOn, asynchronous testing examples.

**Takeaways**:

Angular is evolving fast: Signals, standalone components, and lazy loading improve code efficiency.
Microtasks vs macrotasks mastery is crucial for JS async behavior.
Understanding RxJS, NgRx, interceptors, and authentication is expected even for mid-level roles.
Practical coding skills (closures, promises, arrow functions, prototype methods) remain core to interviews.