Iverson Associates Sdn Bhd (303330-M)
Suite T113 – T114, 3rd Floor, Centrepoint, Lebuh Bandar Utama
Bandar Utama, 47800 Petaling Jaya, Selangor Darul Ehsan
Tel: 03-7726 2678 Fax: 03-7727 9737 Website: www.iverson.com.my

Module Title : **Advanced Angular 21: Modern Architecture, Signals & High-Performance Applications**
Duration : **4 days**

Target Audience

- Angular developers with 1+ year experience
- Frontend engineers transitioning to Angular 17–21
- Developers familiar with RxJS and traditional Angular
- Teams adopting Signals-first and zoneless architecture
- Engineers building scalable enterprise frontend systems

Objectives

By the end of this programme, participants will be able to:

- Design Angular applications using **Signals-first architecture**
- Build applications using **standalone APIs (module-free approach)**
- Implement **zoneless Angular for high-performance rendering**
- Apply **fine-grained reactivity using signal(), computed(), effect()**
- Develop scalable state management using **Signals and NgRx Signals Store**
- Optimize applications using **modern control flow and deferred rendering**
- Build **SSR-enabled Angular applications with hydration support**
- Integrate Angular applications with APIs and external systems
- Apply enterprise-level architectural patterns for maintainability and scalability

Course Outline

Day 1 — Angular 21 Foundations & Modern Architecture

Module 1: Angular Evolution & Modern Mindset

- From NgModules to Standalone architecture
- From RxJS-heavy to Signals-first model
- From Zone.js to zoneless Angular
- Angular 21 key features and direction

Module 2: Standalone Architecture

- Standalone components, directives, and pipes
- bootstrapApplication and app configuration
- Route-based architecture design
- Feature-based folder structure

Module 3: Signals Fundamentals

- signal(), computed(), effect()
- Dependency tracking and reactivity model
- Signal lifecycle and updates
- Signals vs Observables (usage strategy)

Module 4: Modern Templates & Control Flow

- @if, @for, @switch syntax
- @defer for lazy UI loading
- Signal binding in templates
- Rendering performance considerations

Hands-on Lab (Guided)

- Convert NgModule-based app to standalone
- Build components using Signals
- Replace *ngIf / *ngFor with @if / @for
- Implement computed state in UI

Day 2 — Signals Architecture & State Management

Module 5: Signals in Application Architecture

- Local vs shared state
- Service-based signal stores
- Derived state with computed()
- Managing side effects using effect()

Module 6: RxJS vs Signals (Modern Strategy)

- When to use Signals vs Observables
- Interop: toSignal(), toObservable()
- Handling async data and streams

Module 7: Signal Store Patterns

- Lightweight store using Signals
- Immutable update strategies
- Domain-based state organization
- Data flow design

Module 8: NgRx Signals Store

- Overview of NgRx Signals Store
- Signal selectors and computed state
- Handling async operations
- Migration from traditional NgRx

Hands-on Lab (Guided)

- Build a Signal-based store
- Convert RxJS service to Signals
- Implement computed and effects
- Integrate NgRx Signals Store

Day 3 — Performance, Zoneless & Rendering Optimization

Module 9: Zoneless Angular

- Removing Zone.js
- Manual vs reactive updates
- Performance implications

Module 10: Fine-Grained Rendering

- Signals vs traditional change detection
- Minimizing re-renders
- Component design for performance
- Memoization using computed()

Module 11: Deferred & Lazy Rendering

- @defer strategies
- Route-level lazy loading
- Component-level lazy loading
- Progressive rendering

Module 12: Performance Profiling

- Angular DevTools (Signals debugging)
- Identifying bottlenecks
- Optimizing large component trees

Hands-on Lab (Guided)

- Convert app to zoneless mode
- Optimize rendering with Signals
- Implement @defer for lazy UI
- Analyze and fix performance issues

Day 4 — SSR, Integration & Enterprise Design

Module 13: SSR & Hydration

- Angular SSR architecture
- Hydration and pre-rendering
- Signals behavior in SSR
- Avoiding SSR pitfalls

Module 14: API Integration & Data Layer

- HTTP integration with Signals
- Interceptors (auth, logging, errors)
- Error handling strategies
- Caching and retry patterns

Module 15: Web Components Integration

- Angular Elements with standalone components
- Using Signals in Web Components
- Integration with non-Angular systems

Module 16: Enterprise Architecture Patterns

- Feature-based architecture
- Domain boundaries and separation
- Code reuse and shared libraries
- Optional: Nx monorepo strategy
- Optional: Microfrontend concepts

Hands-on Lab (Final Project)

- Implement SSR with signal-based data
- Build API layer with interceptors
- Export Angular component as Web Component
- Apply full architecture best practices
- Present working modern Angular application

Key Takeaways

- Angular 21 adopts a **Signals-first and zoneless architecture**
- Standalone APIs simplify application structure and scalability
- Fine-grained reactivity improves performance significantly
- Modern Angular enables cleaner, more maintainable codebases

- SSR and hydration are essential for production-ready apps
- Proper architecture design ensures long-term scalability and maintainability