**Typescript Micro service course content**

**Duration: 3 Days**

**Prerequisite**

**Participants should have strong knowledge on JavaScript, ECMA standards and Nodejs**

**All the topics are hands on exercises. It is highlighted in yellow color.**

**We will implement to implement microservices patterns using tools**

**All the tools are again highlighted**

**Lab Setup**

**Nodejs 20.x**

**Jet brains Web storm 2022.x / Visual Studio Code**

**Mongodb 4.x / Postgresql**

**Docker Desktop with Kubernetes**

**Day 1 and Day2**

**Introduction**

* Service-based architecture (done)
* From monolithic application to microservice practitioner code examples transition. (done)
* Services-oriented architecture code example. (done)

A screenshot of a computer

Description automatically generated

* Microservices: building blocks (done)
* Microservices: Solver of problems architecture (done)
* Microservices and cloud-native
* Microservices Core Advance Concepts
* Microservices Advanced Concepts

A screenshot of a computer

Description automatically generated with medium confidence

Day 1

**1. Introduction to TypeScript**

**• History and Importance of JavaScript**

**• JavaScript Strengths, Weaknesses**

**• Limitations of JavaScript**

**• Advantages of TypeScript**

**• TypeScript Design Goals**

**• Installing TypeScript**

**• Getting Started**

**2. TypeScript Language Basics**

**• Declaring Variables**

**• Basic Types**

**• Type Annotations**

**• Non-Nullable Types**

**• Basic Data Structures**

**• Operators**

**• Flow Control Statements**

**3. Using Visual Studio Code for TypeScript Development**

**• Tooling Options**

**• VS Code Features**

**• Limitations**

**• Project System**

**• Navigation Basics**

**• Productivity**

**• JavaScript Intellisense**

**• Configuring TypeScript Projects**

**• Compiling TypeScript**

**• Debugging**

**4. Task Automation, Unit Testing, and Continuous Integration**

**• Need for Task Automation**

**• Task Runners**

**• Introduction to Gulp**

**• Using Gulp**

**• Configuring Build Tasks**

**• Debugging Gulp Tasks**

**• Testing Approaches**

**• Testing Frameworks**

**• BDD with Jasmine**

**• Running Tests in a Browser**

**• Writing Tests in TypeScript**

**• Test Runners**

**• Continuous Integration**

**5. TypeScript Type System**

**• Type Compatibility**

**• Type Inference**

**• Access Modifiers**

**• Union Types**

**• Type Guards**

**• Intersection Types**

**• Type Aliases**

**• String Literal Types**

**• Polymorphic “this” Types**

**6. Functional Programming**

**• Functional Programming**

**• Named Functions**

**• Arrow Functions**

**• Function Types**

**• Optional and Default Parameters**

**• Rest Parameters**

**• Function Overloading**

**7. Asynchronous Programming**

**• Importance of Async**

**• Callbacks**

**• Callback Hell**

**• Promises**

**• Generators**

**• Async and Await**

**8. Object-Oriented Programming**

**• Introduction to Object-Oriented Programming**

**• SOLID Principles**

**• Interfaces**

**• Classes**

**• Inheritance**

**• Mixins**

**• Abstract Classes**

**• Class Expressions**

**9. Generics**

**• Algorithm Reuse**

**• Generic constructs**

**• Type Parameters**

**• Constraints**

**• Generic Functions**

**• Generic Interfaces and Classes**

**10. Namespaces and Modules**

**• Name Collisions**

**• Namespaces**

**• Defining Namespaces**

**• Limitations of HTML Script Tags**

**• Module Loaders**

**• ECMAScript 2015 Module Support**

**• Module Organization**

**• Exporting from Modules**

**• Importing from Modules**

**• Module Guidelines**

**Typescript eslint & prettier**

**Day 2**

API 🡪 express/fastify with postgresql

**Splitting Up a Monolith into Microservices**

* Setting up the ecommerce service
* Registering the ecommerce service
* Unregistering a service on shutdown
* Adding the service logic
* Using the microservice from the main app
* Cleaning up the main app
* Creating an image serving endpoint

**Creating a Service Registry (Eureka Server tool hands on)**

Hands on to cover discovery pattern to demonstrate how gateway identifies

available services for Load Balancer and routing

* Setting up the registry
* Setting up endpoints in Express / fastify framework with postgresql
* Sequelize ORM with postgresql
* Registering and deregistering services
* Creating and testing the registration route
* Unregistering services
* Versioning and load balancing
* Querying the registry
* Removing expired services
* Typescript Graphql

**Adding Fault Tolerance and Resilience (hands on using resilience library for retry and redirection 🡪 High availability)**

* What happens if a service fails?
* Understanding circuit breakers
* Building a circuit breaker with Node
* Using the circuit breaker
* Using a cache to bridge outages
* Caching images

**Using Queues with Node.js (Rabbitmq and Kafka tools)**

**(Producer API and Consumer API 🡪 Saga Pattern and Event Sourcing Pattern)**

* Factoring out the feedback service
* Using queues
* Setting up RabbitMQ
* Queuing feedback
* Consuming and storing feedback
* Asynchronous Communications (Kafka), (Rabbitmq)

**Day 3**

* Measuring Performance
* Benchmarking performance
* Benchmarking with ApacheBench
* Benchmarking code with Winston
* Profiling code execution
* Logging and tracing in a microservices architecture (ELK, Jaegar tools)
* Hybrid architectures: Hierarchy and service-based
* Making Architecture Choices
* Testing for microservices
* Secure microservices (JWT) Hands on
* Monitor microservices (Prometheus and Grafana) Hands on
* AWS-SDK for dynamodb (typescript)
* Unit Testing mocha and chai
* Typescript using Jest & sequelize mock.
* Design considerations
* The tradeoffs
* An argument for edge services
* Embracing DevOps (Docker and Kubernetes)
* CI /Continuous Delivery as a requirement
* GitLab/GitHub
* Rancher (Kubernetes) – Harbor/docker hub (Repo)