

Asynchronous, Parallel & Reactive (RxJS) Programming Using TypeScript

Managing time, multiple workers and data streams

Coordinating multiple activities is one of the most difficult areas of advanced application development. In this course we explore all the different options open to developers and see how they can be integrated into modern TypeScript applications.

All modern hardware (even low-end mobile devices) support multiple CPU cores and allow parallel code execution. All have timers that allow asynchronous workloads to be queued for execution in the future. It is up to app developers to exploit the varying capabilities of hardware available to them to deliver optimum apps.

This course is ideal for TypeScript developers who wish to more tightly manage how their code and data are processed - where, when and in what order.

All samples and labs in this course use TypeScript as we think it is best for larger applications – much of what is covered is useful to JavaScript developers too.

We explore code both running on the server (e.g. as part of a Node server application) and in the browser (as standalone TypeScript code, or as part of a larger Angular application).

Contents of One-Day Training Course	
<p>Target Audience TypeScript developers who wish to have more control within their apps over time, multiple workers and observable data streams</p> <p>Prerequisites Understanding of asynchronous and multi-threaded programming from other environments, together with knowledge of TypeScript.</p>	<p>Options for Async & Parallel Review of all the possibilities available to TypeScript developers to manage time, distribute code across contexts to be executed and handle data streaming</p> <p>JavaScript VM Event Loop Need to fully understand the event loop and event ordering to optimize our code Adding events and consuming events</p> <p>Promises & Timers How a promise works: in real life / code Programming with a promise Error handling Using VM timers What setTimeout(0, <func>) means Promises/A+ (https://promisesaplus.com/)</p> <p>TypeScript async/await Making asynchronous source more readable while maintaining capabilities Use of async and await keywords Designing an asynchronous framework</p> <p>Web Workers A web worker is a thread Threads cooperate via message passing Creating and managing web workers Types of web workers Worker lifetime control [Dedicated Shared]WorkerGlobalScope HTML5.3 WindowOrWorkerGlobalScope</p> <p>Messaging Between Workers Message ports / Message parameters PostMessage / onMessage Organizing message flows</p> <p>Shared Array Buffer & Atomics Shared memory within a browser Atomics provide synchronization to protect shared array buffers</p> <p>Zone.js Architecture of Zone.js Multiple zones can live within the same web worker or main browser context Creating and using zones Zone.js is heavily used in Angular - how?</p> <p>RxJS Overview Observable / Observer / Subject Hot & cold next/error/complete Testing RxJS code using jasmine-marbles</p> <p>RxJS Streams An observable as a dual of an enumerable Subscriptions App architecture and stream processing</p> <p>RxJS Operators Processing individual elements via a large collections of operators What's new with pipeable operators Custom operators</p> <p>Advanced RxJS Notifications Schedulers (incl. new TestScheduler) Connectables</p> <p>Project Having explored the fundamental constructs for managing time, code execution and data streams, we conclude with a project that demonstrates all these ideas together.</p>