

Dart Programming

Audience Course Dart Programming

The course Dart Programming is intended for anyone who wants to learn programming in the programming language Dart.

Prerequisites Course Dart Programming

In order to participate in this course basic knowledge of programming in another programming language is beneficial to understanding but is not required.

Realization Training Dart Programming

The theory is discussed on the basis of presentation slides. The theory is explained further through demos. After discussing a module there is the possibility to practice. Course times are from 9.30 to 16.30.

Certification Dart Programming Course

Participants receive an official Dart Programming certificate after successful completion of the course.

Duration: 3 days

Price: € 1850

[Open Schedule](#)



Dart Programming



Content Course Dart Programming

The Dart programming language is a general purpose language originally developed by Google. Dart is open source and now an ECMA standard. This promotes the development of an active ecosystem around the language. The strength of Dart lies mainly in the development of web applications and it is expected that browsers will support Dart directly. Dart is also the foundation of the Flutter Framework for mobile applications.

Dart Intro

In the Dart Programming course participants learn the features of application development with Dart. We discuss the Dart SDK and the transpiler dart2js which generates a JavaScript equivalent of Dart Script.

Dart Syntax

Next attention is paid to data types, generics and control flow in Dart. Dart is type safe, supports type inference and prevents null pointer exceptions with a null safety mechanism.

Functions and Data Structures

The characteristics of functions and data structures in Dart are also discussed. For example data structures from the core library, runes and the mirror system are treated and optional parameters and lambda functions in functions as well.

Classes and Objects

Dart is an object-oriented language and the implementation of constructors, interfaces, exceptions and inheritance in Dart are treated. Extension methods are also covered here.

Concurrency

Finally concurrency in Dart is extensively discussed, whereby parallel tasks can be performed by multiple concurrent threads. Attention is also paid to asynchronous I/O with Dart Futures and the Async Package.

Modules Course Dart Programming

Module 1 : Dart Intro	Module 2 : Language Syntax	Module 3 : Data Structures
What is Dart? Install Dart SDK Dartpad Editor IDE Support Dart to JavaScript dart2js Program Execution Static checker Checked Mode Dart Keywords Dart Identifiers	Type Syntax Numbers Strings Booleans Dynamic Types Final and Const Operators Type Test Operators Iterations with Loops Selections with If Using Labels	Core Library Fixed Length List Growable List List Operations Map Literals Map Constructor Dart Symbols Runes Enumerations String.codeUnits MirrorSystem
Module 4 : Functions and Interfaces	Module 5 : Classes and Objects	Module 6 : Collections and Generics
Function Definition Calling Functions Passing Parameters Return Values Optional Parameters Recursive Functions Lambda Functions Interfaces Implementing Interface Multiple Interfaces	Declaring Classes Fields and Methods Getters and Setters Constructors Named Constructors this Keyword Class Inheritance Types of Inheritance Method Overriding static and super	Dart Collections Set and Queue Iterating Collections Optional Typing Type Safeness Parameterized Types Generic Map Generic List Isolates TypeDefs
Module 7 : Packages	Module 8 : Exceptions	Module 9 : Concurrency
Packaging Programming Units Package Manager pub Package Metadata pubsec.yaml Installing Packages pub get Command Importing Libraries Encapsulation Custom Libraries	Exception Handling try Block on/catch block finally Block ON Block Exception Class Built-in Exceptions Throwing Exceptions Custom Exceptions	Parallel Tasks Multiple Threads Isolates versus Threads Isolate Class spawn Method Dart Futures Async Package Asynchronous I/O readLineSync Method