## Was ist die Programmiersprache Dart?

**Dart** ist eine ECMA-standardisierte Programmiersprache, die hauptsächlich vom Unternehmen Google entwickelt wird.

Als Vielzweck-Programmiersprache entworfen, die eine moderne Alternative zu JavaScript zur Verwendung in Webbrowsern darstellen soll.

Dart soll einige grundsätzliche Probleme von JavaScript überwinden, die sich nach Ansicht der Entwickler von Dart nicht durch eine Weiterentwicklung von JavaScript beheben ließen.

Die Sprache Dart ähnelt etablierten **objektorientierten Programmiersprachen**, zu denen Swift, C# oder Java gehören.

Die **Syntax** ähnelt der Programmiersprache C.

## Geschichte

Die Entwicklung von Dart begann im Jahr 2010 in einem kleinen Team rund um Lars Bak, Gilad Bracha und Kasper Verdich Lund in Aarhus.

Dart wurde auf der GOTO-Konferenz am 10. Oktober 2011 in Aarhus vorgestellt.

Die Veröffentlichung der ersten stabilen Version 1.0 erfolgte am 14. November 2013.

Am 25. Juni 2014 beschloss die Generalversammlung der ECMA mit Dokument „ECMA-408“ die Standardisierung der Sprachspezifikation.

## Laufzeitumgebung

Dart-Programme können im Browser, aber auch auf dem Server ausgeführt werden.

Für Browser ist der Transcompiler Dart2js vorgesehen, der Dartcode nach JavaScript übersetzt.

Die „js Library“ ermöglicht die direkte Verwendung von JavaScript-Code in Dart.

Direkt in der DartVM ausgeführter Dart-Code ist deutlich schneller als vergleichbarer JavaScript-Code.

Bemerkenswert daran ist, dass der dafür verwendete Dart-nach-JavaScript-Compiler selbst in Dart geschrieben ist, und der JavaScript-Output beliebigen gültigen Dartcode in optimiertes JavaScript übersetzt.

Dart-Anwendungen werden mit einem eigenen Mimetype ausgeliefert, nämlich „application/vnd.dart“.

Das Dart-SDK legt derzeit den Schwerpunkt auf clientseitige Entwicklung.

Serverseitige Entwicklung wird beispielsweise mit Bibliotheken wie dart:io unterstützt.

Eine Dart-VM im Browser ist weder in Chrome noch in irgendeinem anderen Browser vorgesehen.

Eine eigenständige Dart-VM wird für Mobile Apps, serverseitige Programme und als Tool zur Unterstützung der Programmierer weiterentwickelt.

Der Fokus der Weiterentwicklung wird auf eine möglichst hohe Produktivität für Programmierer gelegt.

## Entwicklungsziele

Die Entwicklung von Dart wird von folgenden Zielen geleitet:

* Entwicklung einer strukturierten, aber auch flexiblen Programmiersprache für das Web.
* Entwicklung einer leicht erlernbaren, da auf für Programmierer vertrauten Syntax und Techniken beruhenden, Programmiersprache.
* Sicherstellung, dass alle Sprachkonstrukte eine hohe Leistung und einen schnellen Programmstart ermöglichen
* Dart sollte vor allem für Geräte geeignet sein, die mit dem Web kommunizieren, wie Telefone, Tablets, Laptops und auch Server.

## Dart Plattformen

Die flexible Kompilierungstechnologie von Dart gibt Entwicklern die Möglichkeit, Dart Code je nach Verwendungszweck auf verschiedene Art und Weisen zu kompilieren.

### **Dart Native**

Dart Native verwendet sowohl JIT (Just-in-time-Kompilierung) als auch AOT (Ahead-of-time-Kompilierung), um Maschinencode für Zielgeräte wie Smartphones, Desktops oder Server zu entwickeln und auf diesen bereitzustellen.

### **Dart Web**

Bei Dart Web kann ein jeweils eigener Compiler während der Entwicklung (dartdevc) und für die schlussendliche Website (dart2js) verwendet werden.

Wie der Name bereits vermuten lässt, ist Dart Web für die Entwicklung von Web-Applikationen vorgesehen.

## Typisierung

Dart ist optional typisiert. Das bedeutet, dass Dart zwei Runtime-Modi kennt:

* Im *Produktionsmodus* sucht sich der Compiler selbständig die jeweils beste Variante und ignoriert sowohl Typisierungsanweisungen als auch Typisierungsfehler.
* Dieser Modus ist für möglichst effiziente Programmausführung optimiert.
* Im *Checked-Modus* werden die Typanweisungen strikt beachtet und bei Typisierungsfehlern werden Exceptions geworfen.
* Zudem folgt eine Prüfung auf Assertions.
* Die aufwendige Codeanalyse macht diesen Modus langsamer als den *Produktionsmodus*.

Die Vorteile der Typisierungs-Annotationen liegen in der besseren Lesbarkeit für den Menschen und in der deutlich besseren Unterstützung der Programmierer durch Integrierte Entwicklungsumgebungen oder Editoren wie beispielsweise Codevervollständigung, Navigation und frühzeitige Fehlererkennung.

Programmierer, die keinerlei Typisierung einsetzen wollen, können ohne Nachteile für den Produktionsmodus darauf verzichten.

Nur der JavaScript-Output ist gelegentlich nicht so gut optimierbar wie dies mit Typisierungsanweisungen möglich wäre.

## Aufbau

Die Programmiersprache Dart verfügt über Variablen, Operatoren, bedingte Anweisungen, Schleifen, Funktionen, Klassen, Objekte und Aufzählungen.

Sie bietet Vererbung und generische Programmierung als wichtige Konzepte einer objektorientierten Sprache an.

Zum ersten Ausprobieren steht die kostenlose OpenSource-Plattform DartPad zur Verfügung, auf der man sich an die Programmierung mit der Dart-Sprache herantasten kann und auch gleich ein paar **Beispiele** als Drop-down-Menü vorfindet.

Jedes in Dart geschriebene Programm beginnt mit dem Aufruf der „Main“-Funktion.

## Einsatz

Entwicklungen mit der Sprache **Dart** dienen zuallererst Geräten, die **mit dem Internet kommunizieren**, also Smartphones, Tablets und Laptops, aber auch Server.

Vor allem für Programmieranfänger war es bisher wenig realistisch, mobile oder webbasierte Applikationen selbst zu entwickeln.

Der Ansatz der Programmiersprache Dart zielt darauf ab, das Programmieren solcher Apps einfacher zu gestalten.

Googles eigenes Software Development Kit (SDK) [**Flutter**](https://www.ionos.de/digitalguide/websites/web-entwicklung/was-ist-flutter/) ist mit Dart programmiert, ebenso wie das bekannte Werbetool Google Ads.

### Vorteile

Dart ist **Open Source** und somit für jedermann kostenlos zugänglich.

Hinter Dart steht Google, ein großes Unternehmen – das bietet eine **langfristige Perspektive** für die Weiterentwicklung der Sprache.

Dart ist für Programmierer aufgrund ihrer Syntax **leicht erlernbar**.

Die Entwickler haben viele umständliche Begriffswelten anderer Sprachen vereinfacht und intelligent zusammengefasst.

Die Programmiersprache ist **für das Web entwickelt** worden.

Durch direkte (und sehr schnelle) Übersetzung in JavaScript funktioniert Dart auf allen modernen mobilen und Desktop-Browsern.

Zur Entwicklungsarbeit-Arbeit genügt ein **einfacher Texteditor**.

Dazu sind jedoch tiefergehende Kenntnisse der Programmiersprache nötig.

### Nachteile

Die Dart-Programmiersprache ist neu, was zur Folge hat, dass die sich untereinander helfende **Community noch nicht so groß** ist und Lernmaterialien nicht in dem Umfang wie für JavaScript zur Verfügung stehen.

Die **Erstinstallation** eines Editors und des dazugehörigen technischen Hintergrunds auf einem Rechner ist zwar gut dokumentiert, aber **mit Tücken behaftet**.

## Unterschiede

Ein Unterschied zu anderen Programmiersprachen ist, dass Dart **einfach lesbar** ist.

Es gibt **weniger Befehle**, jedoch **mehr Möglichkeiten**.

Bezeichnungen für Variablen kann man frei wählen und somit den selbst geschriebenen Code verständlich halten.

Das spart auch viele zusätzliche Kommentare.

Die Verwendung von Leerzeichen, Tabs und Zeilenumbrüchen ist beliebig gestattet.

Damit können Sie als Programmierer den Code gut strukturieren, was letztlich vom Compiler ignoriert wird.

Ausnahmen sind Schlüsselwörter, Variablen- oder Funktionsnamen – also alle feststehenden Begrifflichkeiten innerhalb der Dart-Sprache.

Dazu gehören z. B. „if“, „else“, „string“, „void“ und weitere.

Letztlich wird auch zwischen Groß- und Kleinschreibung unterschieden, was den Spielraum für Benennungen deutlich erweitert.

# DART-Tutorial

Die **runden Klammern** „()“ deuten auf eine **Funktion** hin.

In den **geschweiften Klammern** „{ … }“ wird der DART-**Code ausgeführt**.

Was hinter den beiden **Schrägstrichen** „//“ steht, ist ein **Kommentar**

### Variablen

Mit Variablen werden feste Größen für das Programm definiert.

Um den **Wert** der Variablen **auszugeben**, kommt der Befehl „**print**“ ins Spiel.

Die **Variablen in DART** nehmen bestimmte **Typen** an.

Das können Ganzzahlen („int“) oder Gleitkommazahlen („double“) sein.

Der Variablentyp „dynamic“ kann im Programmablauf verschiedene Werte und Ausdrücke annehmen.

Die Bezeichnungen für die verwendeten Variablen dürfen keine Schlüsselwörter enthalten.

Auch Zahlen am Anfang sowie Leer- und Sonderzeichen sind nicht zugelassen.

Ausnahmen sind Unterstrich „\_“ und Dollarzeichen „$“.

### Zeichenfolgen

Mit **Zeichenfolgen**, dem Datentyp „**String**“, lassen sich beliebige Zeichen in der DART-Programmierung verarbeiten.

Die Wiedergabe von Text per String erfolgt durch Umschließen des gewünschten Inhalts mit einfachen oder normalen Anführungszeichen.

Wird der Text jedoch mit **dreifach hintereinander angeordneten Anführungszeichen** begonnen und beendet, gibt DART den Text an der umbrochenen Stelle auch als Umbruch aus.

### Zahlen

Dabei können Zahlen oder auch Ausdrücke zusammengerechnet werden.

Um Ergebnisse zu berechnen, nutzt DART **arithmetische Operatoren**.

Es kommen Strings, Gleitkomma- und Ganzzahlen sowie die **Zusammenführung von Elementen** der Programmierung **zu neuen Variablen** vor.

Eine Besonderheit ist beim Zusammenführen von zwei existierenden Variablen in einer neuen Variable zur Ausgabe auf dem Bildschirm zu beachten:

Den **bereits definierten Variablen** wird in diesem Falle ein **Dollarzeichen „$“** vorangestellt.

### Bedingungen stellen

Wenn A eintritt(**if**), erscheint X auf dem Bildschirm.

Wen eintritt B (**elseif**), dann kommt Y;

Wenn beide nicht zutreffen (**else**), wird Z ausgegeben.

### Inkremente und Dekremente

Es handelt sich um das schrittweise Vergrößern oder Verkleinern eines Ausgangswerts.

Im Beispiel wird die Zahl 50 mit den Operatoren „++“ und „--“ verändert.

### Schleifen

Schleifen sind ebenso wichtige Programmroutinen, die immer wieder benötigt werden, um beispielsweise **Vergleiche mit vorhandenen Größen** anzustellen.

Wir verändern den Wert A bis die Größe B erreicht ist.