# Web und Mobile Apps Programmieren mit Dart

Marco Jakob

Kalaidos Fachhochschule Schweiz
majakob@gmx.ch

**Abstract:** Bisher war es kaum realistisch, im Anfängerunterricht mobile oder webbasierte Applikationen zu entwickeln. Die Programmiersprache Dart bietet neue Möglichkeiten, wie solche Applikationen viel einfacher programmiert werden können. Zudem zeigt sich, dass Dart in einigen Bereichen sogar besser geeignet ist in die objektorientierte Programmierung einzuführen als herkömmliche Programmiersprachen.

## 1 Einleitung

Heute sind vor allem *mobile und webbasierte Anwendungen aktuell*. Immer weniger werden Desktop Applikationen, d.h. fest installierte Programme, verwendet. Viele Nutzer bewegen sich fast ausschliesslich im Webbrowser oder auf dem Mobiltelefon.

Es ist deshalb sehr motivierend, wenn Lernende im Unterricht mobile oder webbasierte Anwendungen selber entwickeln können. Zudem können sie solche Anwendungen auf einfache Weise mit ihren Freunden teilen.

*Mit den gängigen Ansätzen*, wie in den letzten Jahren Programmieren unterrichtet wurde, ist es aber *kaum oder nur sehr beschränkt möglich*, mobile oder webbasierte Anwendungen zu entwickeln.

Im Folgenden wird zuerst dargelegt, weshalb es mit herkömmlichen Möglichkeiten schwierig ist, mobile oder webbasierte Applikationen im Anfängerunterricht zu programmieren. Anschliessend wird die neuere Programmiersprache *Dart* vorgestellt und untersucht, in wiefern sich diese für den Unterricht eignet.

## 2 Problematik mobiler Anwendungen

Android und Apples iOS sind die gängigsten Plattformen für mobile Anwendungen. Der Einsatz dieser Plattformen im Unterricht ist mit Schwierigkeiten verbunden, die hier kurz erwähnt werden.

**iOS-Apps.** Die Entwicklung von iOS-Applikationen erfolgt in einer eigenen Sprache namens Objective-C speziell für die iOS-Plattform. Dies bedeutet, dass die Programme auch nur auf iPhones bzw. iPads laufen. Zum Entwickeln von iOS-Anwendungen benötigt man

(bevorzugt) ein Mac-Betriebssystem. So ein System dürfte nicht überall zur Verfügung stehen. Zudem ist es sehr umständlich bis unmöglich für eine Schule, die eigenen Programme über den App Store auf anderen iOS-Geräten zu installieren.

**Android-Apps.** Obwohl Android-Applikationen in Java programmiert werden, stellt sich die gleiche Problematik wie bei iOS-Applikationen: Die Programme laufen nur auf Geräten mit einem Android-Betriebssystem. Der Besitz eines Android-Gerätes kann aber nicht vorausgesetzt werden. So werden Lernende ausgeschlossen, die ihr Programm auf einem iOS-Gerät, einem Windows Phone oder auf dem Desktop ausführen möchten.

## 3 Problematik webbasierter Anwendungen

Webbasierte Anwendungen haben den Vorteil, dass sie *sowohl auf Desktopbrowsern* (Internet Explorer, Firefox, Chrome, Safari) *als auch auf mobilen Browsern* laufen können. Zur Entwicklung von Webanwendungen gibt es konvenzionell zahlreiche Möglichkeiten, die aber alle ihre Tücken haben, wenn sie im Unterricht eingesetzt werden sollen.

**PHP, ASP.net etc.** Diese sogenannten Skriptsprachen werden häufig verwendet, um dynamische Webanwendungen zu erstellen. Dabei werden auf einem Webserver HTML-Seiten generiert, welche dann im Browser angezeigt werden können. Die so notwendige Kommunikation zwischen dem Server und dem Browser auf dem Client bringt zusätzliche Komplexität in das Erlernen einer Programmiersprache. Ausserdem eignen sich Skriptsprachen nicht besonders gut, um moderne, objektorientierte Programmierkonzepte zu unterrichten.

**Python, Java, C++, Visual Basic etc.** Mit all diesen Sprachen lassen sich objektorientierte Programmierkonzepte vermitteln. Der Weg zu einer Webapplikation ist jedoch sehr aufwändig und anspruchsvoll. All diese Programmiersprachen laufen nicht direkt im Browser. Somit muss, wie bei den Skriptsprachen, eine Serverapplikation geschrieben werden, welche HTML-Seiten generiert. Und da diese Sprachen nicht für die Webprogrammierung entwickelt wurden, ist die Komplexität noch höher als bei den oben genannten Skriptsprachen.

**JavaScript.** JavaScript ist die einzige Programmiersprache, welche von allen gängigen Browsern unterstützt wird. Es ist somit möglich, damit ganze Webanwendungen zu schreiben, welche direkt in Desktopbrowsern und mobilen Browsern laufen können. Leider ist aber JavaScript für den Unterricht weniger geeignet. JavaScript ist eine Programmiersprache mit schwierigen Konzepten und vielen Ausnahmefällen. Diese Hindernisse sind (nicht nur) für Anfänger sehr frustrierend.

# 4 Dart

Google hat eine neue Sprache namens Dart[1] entwickelt.

Dart ist *für das Web gemacht* und lässt sich direkt in JavaScript übersetzen. Somit laufen Dart-Anwendungen ohne zusätzliche Installation in allen modernen Desktopbrowsern und mobilen Browsern.


Abbildung 1: Dart-Programme laufen auf allen modernen Plattformen
*(Bildquelle: Sean Poon und Design Contest)*

Dart ist eine *objektorientierte Sprache*, welche sich sehr gut für den Unterricht eignet. Sie ist stark an die bekanntesten Programmiersprachen wie Java und C++ angelehnt, ist aber konsistenter und eleganter. Das macht die Sprache einfach zu lernen für Umsteiger und viel einfacher für Programmieranfänger (siehe Code-Beispiele weiter unten).

Fast genauso wichtig wie die Programmiersprache selbst ist das, was rund um die Sprache zur Verfügung steht:

*Dart Editor* ist eine komfortable Entwicklungsumgebung für Dart. Der Editor enthält professionelle Programmierhilfen wie Code-Vervollständigung, Umbenennen von Variablen und Debugging. Trotzdem ist der Dart Editor sehr einfach und übersichtlich gehalten, d.h. er enthält keine unnötigen Elemente, die Anfänger abschrecken könnten.

Der Dart Editor läuft *ohne Installation* auf Windows, Linux und Mac. Da keine Installation nötig ist, kann er zum Beispiel auch von einem USB-Stick gestartet werden.

Viele *nützliche Bibliotheken* sind bei Dart schon eingebaut. Zum Beispiel für mathematische Berechnungen, für Interaktionen mit HTML, zum Speichern in Dateien und Datenbanken, für Kryptografie etc. Zusätzliche Bibliotheken werden nach Bedarf automatisch von einem zentralen Server heruntergeladen.

[1] Siehe http://www.dartlang.org

Es gibt drei Möglichkeiten, Dart-Programme auszuführen: In der Kommandozeile, als JavaScript in allen modernen Browsern oder in Chromium, einem Google Browser mit Dart Virtual Machine.

Man kann auch *Server-Applikationen* in Dart programmieren. So ist es möglich, eine gesamte Client-Server-Applikation in der gleichen Sprache zu verfassen. Der Server wird in der Kommandozeile gestartet und von einem Browser Client angesprochen. Mit kaum einer anderen Programmiersprache ist dies möglich.

## 5 Dart Code-Beispiele

Anhand von Code-Beispielen soll gezeigt werden, wie einfach es ist, mit Dart zu programmieren. Wer bereits eine Programmiersprache kennt, dem wird Dart sehr vertraut vorkommen. Dies ermöglicht den Umstieg auf diese Sprache innert weniger Stunden.

Dort, wo herkömmliche Sprachen unnötige Stolpersteine beinhalten, bietet Dart elegante Lösungen. Um dies aufzuzeigen, werden bei einigen Beispielen Parallelen zu Java gezogen.

### 5.1 Hello World

Mit diesem Klassiker zeigt sich, wie aufwändig es ist, ein erstes Programm in einer Sprache zum laufen zu bringen. Ein minimales Programm in Dart sieht wie folgt aus:

```
main() {
  print("Hello, World!");
}
```

Angenommen, das Programm oben wurde unter *helloworld.dart* gespeichert. Nun kann es von der Kommandozeile wie folgt gestartet werden:

```
dart helloworld.dart
```

**Kommentar.** Obwohl in Dart Klassen häufig verwendet werden, gibt es die Möglichkeit, Funktionen ohne Klassen zu definieren. In Java ist dies nicht möglich. Dort würde ein entsprechendes Programm wie folgt aussehen:

```
// Java
public class HelloWorld {

  public static void main(String[] args) {
    System.out.println("Hello, World!");
  }
}
```

Der Anfänger ist mit einem solchen Programm schnell überfordert. Neben den Schlüsselbegriffen *public, class, static* werden fortgeschrittene Sprachelemente wie Arrays und ein statischer Aufruf verwendet. Um dieses Java *Hello, World* zu verstehen, braucht es bereits etliches an Programmierkenntnissen.

Ein weiterer Schritt, der bei Dart wegfällt, ist das Kompilieren. Kompilieren ist für die Dart Virtual Machine[2] (VM) nicht nötig, da die VM direkt den Quellcode ausführt.

### 5.2 Funktionen und Variablen

```
// Funktion definieren.
printNumber(int number) {
  // Nummer auf Konsole ausgeben.
  print("Die Nummer ist $number.");
}

main() {
  // Variable deklarieren und initialisieren.
  var number = 42;
  // Funktion aufrufen.
  printNumber(number);
}
```

**Kommentar.** Typen von Variablen, Parametern und Rückgabewerten können optional angegeben werden. Man könnte also bei *var number* den Typ genauer angeben und *int number* schreiben. Damit würde man vom Dart Editor gewarnt, wenn man etwas anderes als einen Integer in diese Variable speichern möchte. Deshalb ist es oft sinnvoll, den Typ anzugeben. Für eine Einführung in die Thematik der Variablen ist es aber praktisch, wenn man zuerst ohne Typen arbeiten kann. Diese Möglichkeit hat man bei den meisten Programmiersprachen nicht.

### 5.3 Integer

```
// String in einen int umwandeln.
int i = int.parse("5");

// int in einen String umwandeln. Da 22 ein Objekt ist,
// kann man Funktionen darauf aufrufen.
String s = 22.toString();
```

[2] Gründe, warum Dart VM keine Bytecode VM ist: http://www.dartlang.org/articles/why-not-bytecode/

**Kommentar.** In Dart gibt es keine Unterscheidung zwischen elementaren Datentypen und Objekten (wie etwa in Java). Alles, was in eine Variable gespeichert werden kann, ist ein Objekt. Das ist für das Verständnis viel einfacher!

Ausserdem kann ein int eine beliebige Grösse haben. Man braucht sich also nicht darum zu kümmern, ob man noch im Zahlenbereich drin ist oder wie in Java zwischen int und long unterscheiden.

### 5.4 Klassen

```
import 'dart:math';

class Point {
  num x;
  num y;

  // Konstruktor (kurze Schreibweise)
  Point(this.x, this.y);

  num distanceTo(Point other) {
    var dx = x - other.x;
    var dy = y - other.y;
    return sqrt(dx * dx + dy * dy);
  }
}

main() {
  var p = new Point(2, 3);
  var q = new Point(3, 4);

  // String Interpolation
  print('Distanz von p zu q = ${p.distanceTo(q)}');
}
```

**Kommentar.** Definition einer Klasse und das Erzeugen von Objekten funktionieren analog zu Java. Einzig der Konstuktor mit den Zuweisungen zu den Variablen ist eine verkürzte Schreibweise. Optional könnte man den Konstruktor auch gleich schreiben wie in Java:

```
// Konstruktor (lange Schreibweise)
Point(num x, num y) {
  this.x = x;
  this.y = y;
}
```

### 5.5 Interaktion mit dem Browser

```
import 'dart:html';

main() {
  // HTML Knopf erstellen.
  var button = new ButtonElement()
  ..text = 'Bestellen'
  ..classes.add('wichtig')
  // Beim Klicken die Funktion handleOnClick aufrufen.
  ..onClick.listen(handleOnClick);

  // Knopf in HTML einfuegen.
  query('#bestellung').children.add(button);
}

void handleOnClick(MouseEvent event) {
  window.alert('Danke!');
}
```

**Kommentar.** Graphische Benutzeroberflächen werden in Dart für den Browser geschrieben. Nach dem Übersetzen des Dart-Codes in JavaScript können alle modernen Browser das Dart-Programm anzeigen. In Java würde man typischerweise eine Benutzeroberfläche mit Swing oder JavaFX programmieren. Im Gegensatz zu Dart lassen sich diese aber nicht direkt im Browser ausführen.

## 6 Fazit

Mit Dart gibt es erstmals eine realistische Möglichkeit, mobile und webbasierte Applikationen im Anfängerunterricht zu programmieren. Zudem zeigt sich, dass Dart in einigen Bereichen sogar besser geeignet ist in die objektorientierte Programmierung einzuführen als herkömmliche Sprachen. Dies ist möglich, da Dart eine neuere Sprache ist und deshalb auf jahrzentelanger Erfahrung mit Sprachen wie Java, C++, Visual Basic, etc. aufbauen konnte.

## 7 Weitere Details

Unterrichtsmaterial und weitere Informationen zum Programmieren mit Dart finden Sie auf der folgenden Webseite:

- http://edu.makery.ch (unter *Projects*, *Learn Dart*)