

▼ Dart für Java-Programmierer

Thomas Künneth
MATHEMA Software GmbH

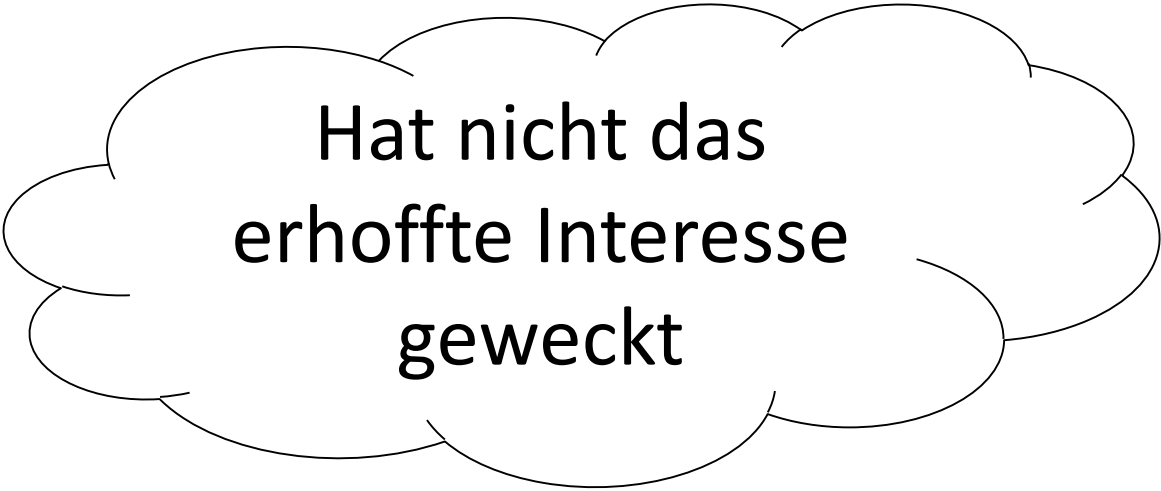
- ▼ Objektorientierte Allzweck-Programmiersprache mit Einfachvererbung
- ▼ Ausführung...
 - ▼ Stand alone
 - ▼ Im Browser
 - ▼ Android und iOS
- ▼ Wird seit 2010 von Google entwickelt
- ▼ Ende 2013 erste stabile Version 1.0
- ▼ Seit 2014 ECMA-standardisiert

- ▼ Gleichberechtigte Alternative zu JavaScript
- ▼ Browser (Chrome) sollte Dart-Laufzeitumgebung enthalten

○

○

○



Hat nicht das
erhoffte Interesse
geweckt

- ▼ Konzentration auf Übersetzung nach JavaScript
 - ▼ War von Beginn an möglich (`dart2js`)
 - ▼ Wird seitdem kontinuierlich erweitert (`dartdevc`)
- ▼ Integrierbarkeit in riesiges JavaScript-Ökosystem
 - ▼ Nutzung etablierter Frameworks
 - ▼ Einbettung in etablierte Abläufe und Toolketten

The screenshot shows the DartPad web application in a browser window. The browser's address bar displays the URL `https://dartpad.dartlang.org`. The application interface includes a top bar with the DartPad logo, a 'New Pad...' button, a 'Reset...' button, and a title 'dry-morning-1663'. On the right side of the top bar are 'Share...' and 'Samples' buttons. Below the top bar, there are tabs for 'DART', 'HTML', and 'CSS', with 'DART' being the active tab. A 'Run' button is located to the right of the code editor. The code editor contains the following Dart code:

```
void main() {  
  for (int i = 0; i < 5; i++) {  
    print('hello ${i + 1}');  
  }  
}
```

To the right of the code editor is a panel with two tabs: 'HTML OUTPUT' and 'CONSOLE'. The 'CONSOLE' tab is active, displaying the output of the code:

```
hello 1  
hello 2  
hello 3  
hello 4  
hello 5
```

At the bottom of the application, there is a footer with links for 'Privacy policy' and 'Send feedback' on the left, and a 'Strong mode (What's this?)' checkbox on the right.



The screenshot shows an IDE window titled "Demo.dart - Demo - [~/Entwicklung/IntelliJ/Demo]". The editor displays the following Dart code:

```
1 void main() {  
2   for (int i = 0; i < 5; i++) {  
3     print('hello ${i + 1}');  
4   }  
5 }
```

The left sidebar shows the "Structure" view with a single entry: `main() → void`. The bottom "Run" panel shows the execution output for "Demo.dart" using the "OBSERVATORY" tool:

```
dart Demo.dart [Observatory: http://127.0.0.1:55548/]  
hello 1  
hello 2  
hello 3  
hello 4  
hello 5  
.....  
Process finished with exit code 0
```

The status bar at the bottom indicates "Process finished with exit code 0" and "10:1 LF UTF-8".

- ▼ Dart SDK: <https://www.dartlang.org/install>
- ▼ Dartium (optional): <https://www.dartlang.org/install/archive>
- ▼ IDE-spezifisches Plugin

```
1 main() {  
2  
3   for (int i = 1; i <= 3; i++) {  
4  
5     if (i < 2) {  
6       continue;  
7     }  
8  
9     print("Hello Dart $i");  
10  }  
11 }
```

Top-Level-Funktion

Schleife

Bedingung

Funktionsaufruf

String Interpolation

```
Hello Dart 2  
Hello Dart 3
```


- ▼ `num` abstrakte Oberklasse
- ▼ `int` mathematische Ganzzahlen
- ▼ `double` 64 Bit Fließkomma nach IEEE-754

DartPad

New Pad...

Reset...

broa...

Share...

Samples

[DART](#)

HTML

CSS

▶ Run

```
import "dart:math";

void main(List<String> args) {
  var i = pow(2, 53);
  print(i);
  print(i + 1);
  i *= -1;
  print(i);
  print(i - 1);
}
```

HTML OUTPUT

[CONSOLE](#)

9007199254740992

9007199254740992

-9007199254740992

-9007199254740992

9007199254740992

9007199254740993

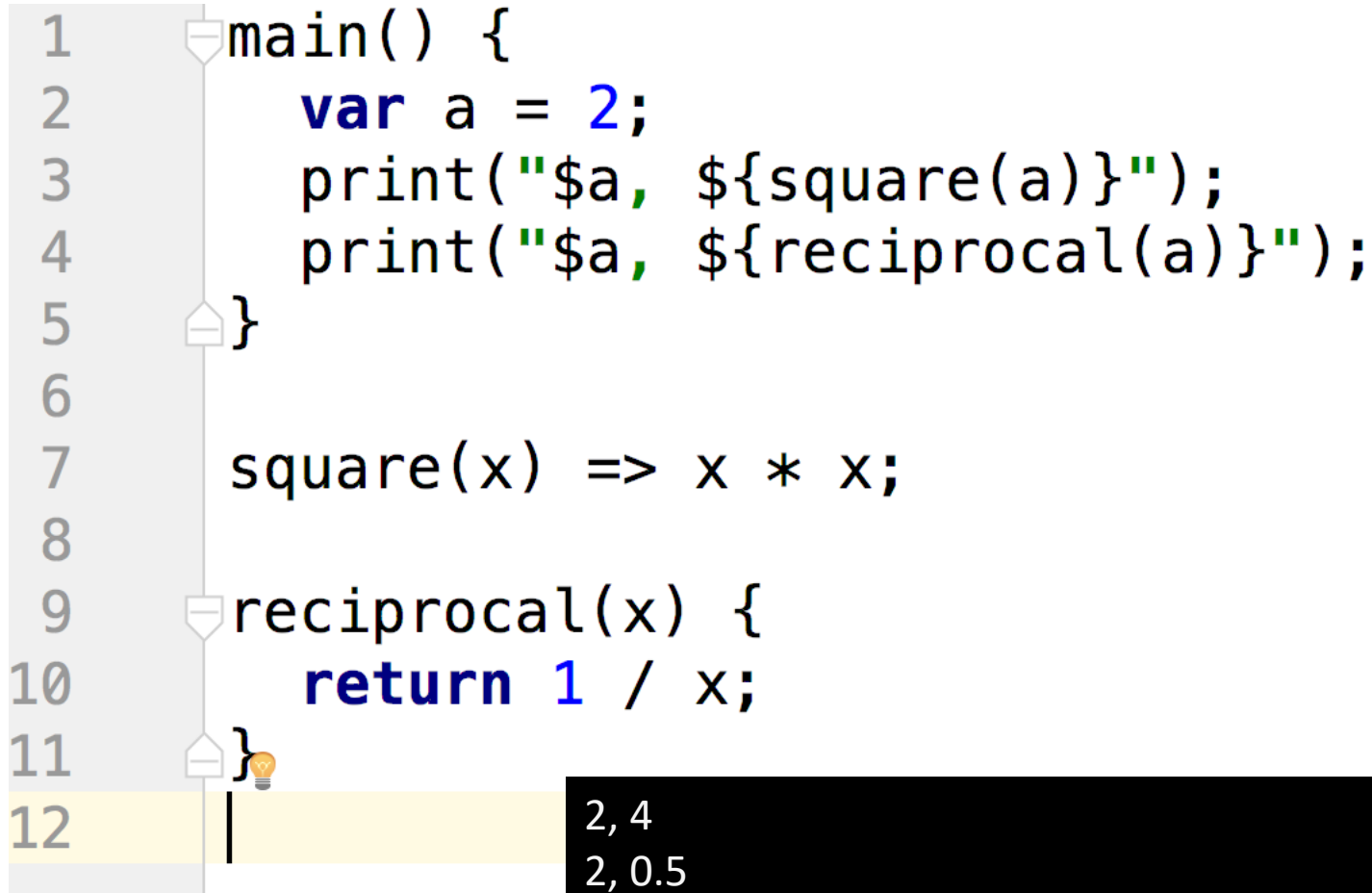
-9007199254740992

-9007199254740993

MATHEMA

- ▼ Zeichenketten: UTF-16 (`String`)
- ▼ Runen zur Darstellung von UTF-32-Codeeinheiten in Zeichenketten (`\u{1f600}`)
- ▼ Wahrheitswerte (`bool`)
- ▼ Listen (`[1, 2, 3]`)
- ▼ Maps (`{"answer" : 42}`)

```
1  main() {  
2      var a = 2;  
3      print("$a, ${square(a)}");  
4      print("$a, ${reciprocal(a)}");  
5  }  
6  
7      square(x) => x * x;  
8  
9  reciprocal(x) {  
10     return 1 / x;  
11 }  
12
```



2, 4
2, 0.5

```
1  main() {  
2  
3      var a = 10;  
4      print(a + 0.5);  
5      a = "Hallo";  
6      print(a);  
7  
8      final double b = .0;  
9      print(b);  
10 }
```

ohne Typannotation

mit Typannotation

```
10.5  
Hallo  
0.0
```

- ▼ Fehlt die Typannotation, wird `dynamic` verwendet
- ▼ `final` kann nicht in Verbindung mit `var` verwendet werden
- ▼ In diesem Fall stattdessen: `const a = 10;`
- ▼ Zulässige Operationen ergeben sich aus dem zugewiesenen Objekt

```
1  main() {  
2      var a = new Object();  
3      print(a.runtimeType);  
4  
5      var b = 1;  
6      print(b.runtimeType);  
7      print(b.runtimeType == Object);  
8      print(b is Object);  
9  }
```

```
Object  
int  
false  
true
```

- ▼ Alles in Dart ist ein Objekt
- ▼ Alles ist eine Instanz einer Klasse
- ▼ Alle Klassen leiten von `Object` ab

- ▼ Beispiele:
 - ▼ `null.runtimeType` liefert `Null`
 - ▼ `42.runtimeType` liefert `int`
 - ▼ `123.45.runtimeType` liefert `double`

```
1 main() {  
2     var a = "Hallo";  
3     var b = " Hallo".substring(1);  
4  
5     print(a == b);  
6     print(identical(a, b));  
7 }
```

true
false


```
1  void main() {  
2      var b1 = new Book("Android 7",  
3          "978-3-8362-4200-4");  
4  
5      print("${b1.title}, ${b1.isbn}");  
6  }  
7  
8  class Book {  
9      var title;  
10     var isbn;  
11  
12     Book(this.title, this.isbn);  
13 }
```

Titel: Android 7, ISBN: 978-3-8362-4200-4

```
1  main() {  
2      var a = new C();  
3      a.a = 42;  
4      a.b = 24;  
5      print("${a.a}, ${a.b}");  
6  }  
7  
8  class C {  
9      var _b;  
10  
11      get a => 123;  
12      set a(wert) => {};  
13  
14      get b => _b;  
15      set b(val) => _b = 321;  
16  }
```

123, 321

Demo.dart - Demo - [~/Entwicklung/IntelliJ/Demo]

Structure

main() → void

```

1 void main() {
2   int a;
3   assert(a != null);
4 }
5

```

Run Demo.dart

dart Demo.dart [Observatory: http://127.0.0.1:55939/]

Unhandled exception:

'file:///Users/thomas/Entwicklung/IntelliJ/Demo/Demo.dart': Failed assertion: line 3 pos 10: 'a != null' is not true.

#0 _AssertionError._throwNew (dart:core-patch/errors_patch.dart:24)

#1 _AssertionError._checkAssertion (dart:core-patch/errors_patch.dart:31)

#2 main (file:///Users/thomas/Entwicklung/IntelliJ/Demo/Demo.dart:3:10)

#3 _startIsolate.<anonymous closure> (dart:isolate-patch/isolate_patch.dart:261)

#4 _RawReceivePortImpl._handleMessage (dart:isolate-patch/isolate_patch.dart:148)

Process finished with exit code 255

Process finished with exit code 255

46 chars, 5 lines 5:1 LF UTF-8

▼ Checked Mode (entwicklerfreundlich)

- ▼ Aktivierung durch `--checked`
- ▼ Assertions sind aktiv
- ▼ Typinkonsistenzen verursachen Exceptions

▼ Production Mode (auf Geschwindigkeit getrimmt)

- ▼ Assertions sind deaktiviert
- ▼ Typannotationen werden nicht beachtet: `var a = 42;` und `int a = 42;` haben zur Laufzeit dieselbe Bedeutung
- ▼ `String e = 1 + 2;` erzeugt keinen Fehler

- ▼ Klassenbasierte, objektorientierte Programmiersprache mit Einfachvererbung
- ▼ C-artige Syntax
- ▼ Optional typisiert

- ▼ Strong Mode (restriktiveres Typsystem)
- ▼ `async` und `await`
- ▼ Mixins und implizite Interfaces
- ▼ Isolates
- ▼ Sichtbarkeit
- ▼ Importe, Libraries und Packages

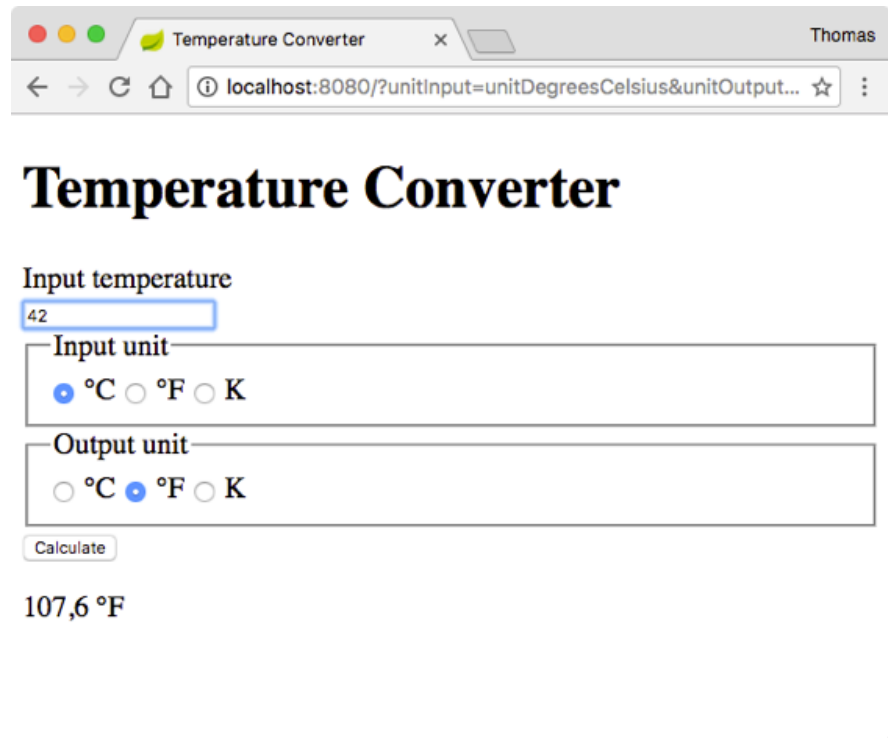
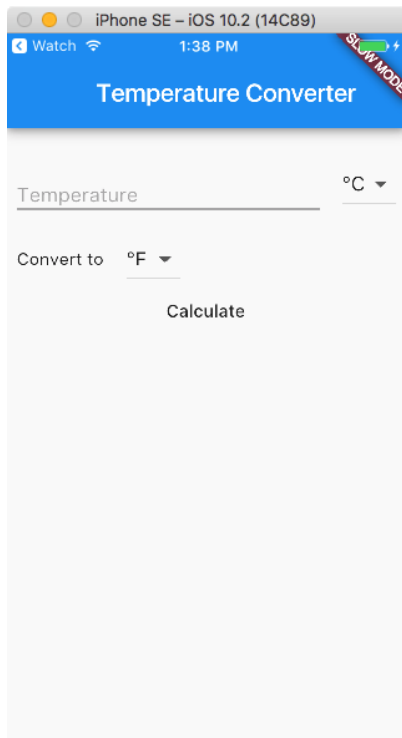
- ▼ „klassische“ Konsolen- und Serveranwendungen
- ▼ Webanwendungen
 - ▼ Unter Verwendung von `dart:html`
 - ▼ [AngularDart](#) – Angular 2 für Dart
 - ▼ [Polymer Dart](#) – Dart-Anwendungen mit Polymer
- ▼ Apps für Android und iOS mit [Flutter](#)

Webanwendung mit `dart:html`

<https://github.com/tkuenneth/darttalks/tree/master/TemperatureConverter>

Mobile App mit Flutter

<https://github.com/tkuenneth/darttalks/tree/master/TemperatureConverter-Flutter>



- ▼ Java-Entwickler finden viel Vertrautes
- ▼ Optionale Typisierung ist zu Beginn gewöhnungsbedürftig
- ▼ Macht Spaß, Dart-Programme zu schreiben:
 - ▼ Sehr gute Tool-Unterstützung
 - ▼ Schlanker Code
 - ▼ Weniger oft Fehlermeldungen des Compilers

Vielen Dank

thomas.kuenneth@mathema.de

Twitter: [@tkuenneth](https://twitter.com/tkuenneth)

Blog: <http://kuennetht.blogspot.de/>