

```
main() {  
    print("Hello Topconf");  
}
```

<https://github.com/tkuenneth/darttalks>

▼ Let's play Dart

Thomas Künneth
MATHEMA Software GmbH

- ▼ Object-oriented general purpose language with single inheritance and mixins
- ▼ Execution environments
 - ▼ Standalone
 - ▼ Browser
 - ▼ Android and iOS (since end of 2015)

- ▼ Developed by Google since 2010
- ▼ End of 2013 first stable version 1.0
- ▼ ECMA standard since 2014
- ▼ Initially planned as a genuine alternative to JavaScript
 - ▼ Browser (Chrome) would contain a Dart runtime environment
 - ▼ Did not work out as expected (not much impact)
- ▼ Spring 2015: new focus
 - ▼ No competition with but integration in the JavaScript ecosystem
 - ▼ More focus on transpilation to JavaScript: `dart2js` gets a more capable brother `dartdevc`
- ▼ End of 2015: Flutter

Try it online (001.dart)

4

The screenshot shows the DartPad web application in a browser. The address bar shows the URL <https://dartpad.dartlang.org>. The interface includes a top bar with the DartPad logo, buttons for 'New Pad...', 'Reset...', and 'Format', a file name 'falling-truth-3564', and buttons for 'Share...' and 'Samples'. Below this is a tabbed interface with 'DART', 'HTML', and 'CSS' tabs. The 'DART' tab is active, displaying the following code:

```
main() {  
  for (var i = 0; i <= 3; i++) {  
    print(printer(i));  
  }  
}  
  
printer(value) {  
  return "$value x $value = ${calc(value)}";  
}  
  
calc(value) => value * value;
```

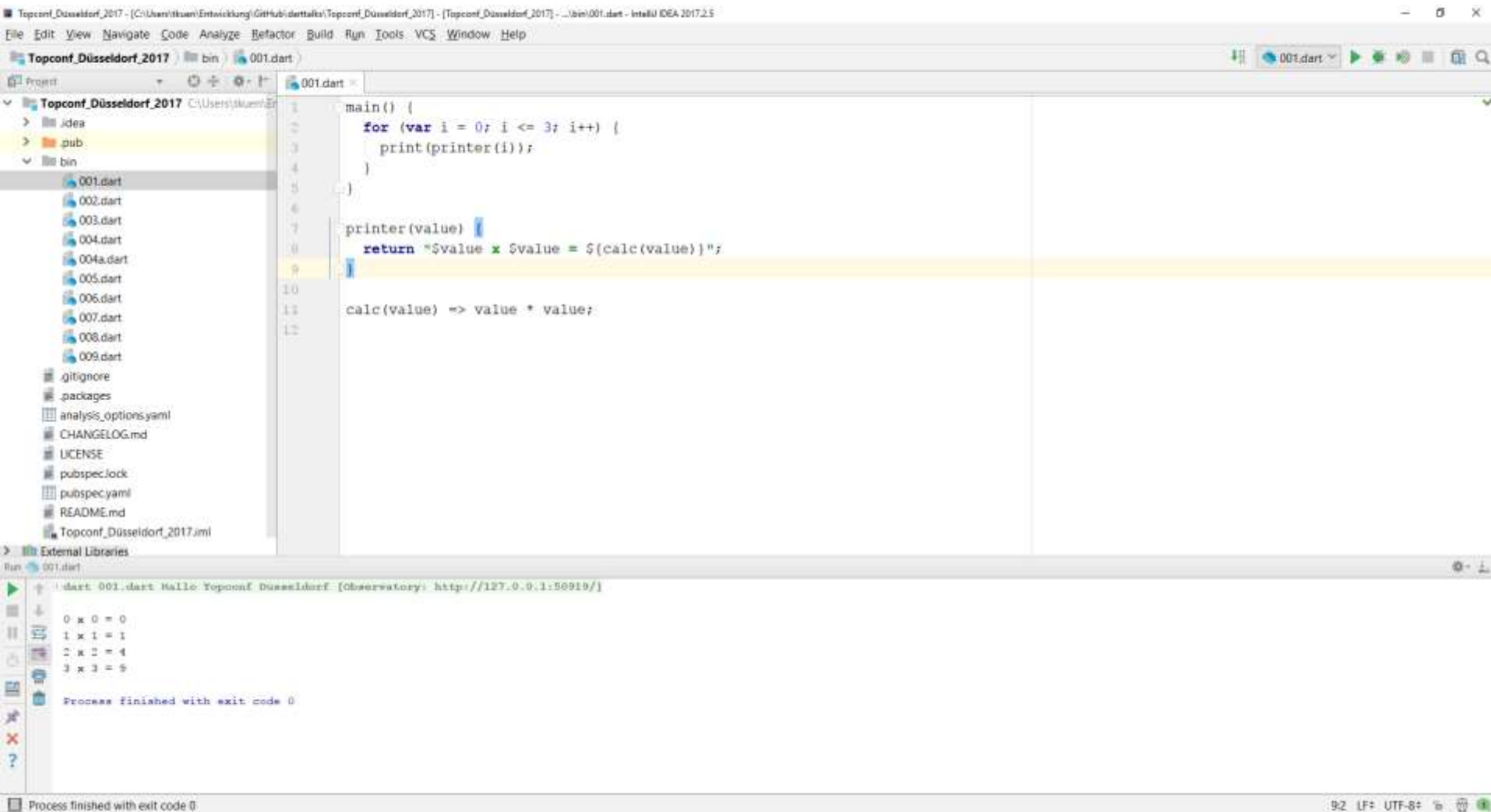
A 'Run' button is located to the right of the code editor. The output panel on the right shows the results of the code execution:

```
0 x 0 = 0  
1 x 1 = 1  
2 x 2 = 4  
3 x 3 = 9
```

At the bottom of the interface, there are links for 'Privacy policy' and 'Send feedback', and a checkbox for 'Strong mode (what's this?)' which is currently checked.

Or use your favorite IDE (001.dart)

5



The screenshot shows an IDE window titled "Topconf_Düsseldorf_2017 - [C:\Users\skuen\Entwicklung\Github\darttalks\Topconf_Düsseldorf_2017] - [Topconf_Düsseldorf_2017] - ...bin\001.dart - IntelliJ IDEA 2017.2.5". The menu bar includes File, Edit, View, Navigate, Code, Analyze, Refactor, Build, Run, Tools, VCS, Window, and Help. The toolbar shows icons for running and debugging. The "Project" view on the left shows the project structure for "Topconf_Düsseldorf_2017", including folders like "bin" and "pub", and files like "001.dart", "002.dart", "003.dart", "004.dart", "004a.dart", "005.dart", "006.dart", "007.dart", "008.dart", "009.dart", "gitignore", "packages", "analysis_options.yaml", "CHANGELOG.md", "LICENSE", "pubspec.lock", "pubspec.yaml", "README.md", and "Topconf_Düsseldorf_2017.xml". The "001.dart" file is selected, and its content is displayed in the editor:

```
1 main() {  
2   for (var i = 0; i <= 3; i++) {  
3     print(printer(i));  
4   }  
5 }  
6  
7 printer(value) {  
8   return "$value x $value = ${calc(value)}";  
9 }  
10  
11 calc(value) => value * value;  
12
```

The "Run" view at the bottom shows the execution output for "001.dart":

```
dart 001.dart Hallo Topconf Düsseldorf [Observatory: http://127.0.0.1:50919/]  
0 x 0 = 0  
1 x 1 = 1  
2 x 2 = 4  
3 x 3 = 9  
Process finished with exit code 0
```

The status bar at the bottom indicates "Process finished with exit code 0" and "9:2 LF UTF-8".

- ▼ Runs on macOS, Windows and Linux
- ▼ Dart SDK: <https://www.dartlang.org/install>
- ▼ IDE-specific plugin

```
1 void main(List<String> args) {  
2     if (args.length < 1) {  
3         print("sorry, no args passed");  
4     } else {  
5         for (var arg in args) {  
6             print(arg);  
7         }  
8     }  
9 }
```

- ▼ `num` abstract base class
- ▼ `int` mathematical integers
- ▼ `double` 64 bit floating point (IEEE-754)
- ▼ **Strings:** UTF-16 (`String`)
- ▼ **Runes** to display UTF-32 code points in strings (`\u{1f600}`)
- ▼ **Booleans** (`bool`)
- ▼ **Lists / arrays** (`[1, 2, 3]`)
- ▼ **Maps** (`{"answer" : 42}`)


```
1  import "dart:math";
2
3  void main(List<String> args) {
4      var i = pow(2, 53);
5      print(i);
6      print(i + 1);
7      i *= -1;
8      print(i);
9      print(i - 1);
10 }
```

```
1  main() {  
2      print(function1(1, b: 2, c: 3));  
3      print(function1(1, c: 3));  
4      // print(function1(1, b: 2));  
5  }  
6  
7  function1(num a, {num b = 42, num c}) {  
8      print("a = $a");  
9      print("b = $b");  
10     print("c = $c");  
11     return a + b + c;  
12 }
```

```
1  main() {  
2      // functions can be assigned to variables  
3      var f1 = (x) => x * x;  
4      print(f1(4));  
5      // and they can be anonymous  
6      var l = [1, 2, 3];  
7      l.forEach((val) {  
8          print("$val * $val = ${val * val}");  
9      });  
10 }
```

```
1  main() {  
2      var a = 1;  
3      dynamic b = 0.5;  
4      print(a + b);  
5      b = "Hallo";  
6      print(b);  
7  }
```

▼ Production Mode (speed)

- ▼ Type annotations are ignored: `var a = 42;` and `int a = 42;` have the same meaning during runtime
- ▼ `String e = 1 + 2;` causes no error

▼ Checked Mode (developer friendly)

- ▼ Activate with `--checked`
- ▼ Type inconsistencies cause Exceptions

▼ Strong Mode

- ▼ Makes the Dart type system sound
- ▼ Becomes statically typed, yet often types can be omitted
- ▼ Will be standard in Dart 2.0

```
1  main() {  
2      var a = new Object();  
3      print(a.runtimeType);  
4      var b = 1;  
5      print(b.runtimeType);  
6      dynamic c;  
7      print(c.runtimeType);  
8      print(null.runtimeType);  
9      print(true.runtimeType);  
10     print(123.45.runtimeType);  
11 }
```

- ▼ Everything in Dart is an object
- ▼ During runtime everything is the instance of a class
- ▼ All classes are derived from `Object`

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

```
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 }
```



```
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  }
```

- ▼ A class that usually extends `Object`, ...
- ▼ ... declares no constructors
- ▼ ... should not call `super`

- ▼ Are referenced with the `with` keyword

```
1  main() {  
2      var b = new B();  
3      b.sayHelloTo("Topconf");  
4      print(b.magic);  
5  }  
6  
7  class A {  
8      sayHelloTo(name) {  
9          print("Hello, $name");  
10     }  
11 }  
12  
13 class B extends Object with A {  
14     int magic = 123;  
15 }
```

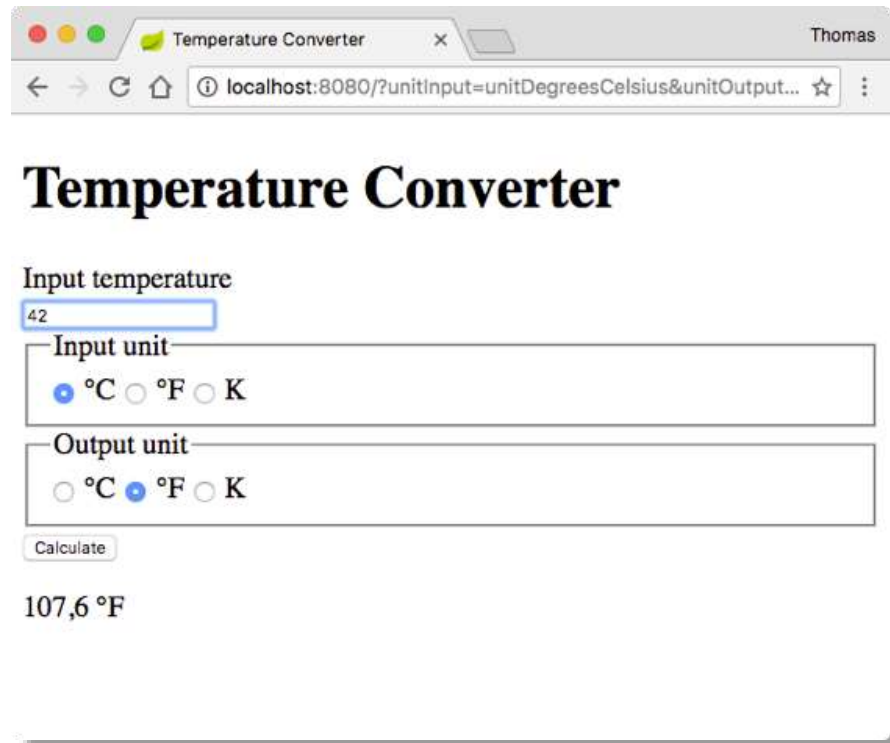
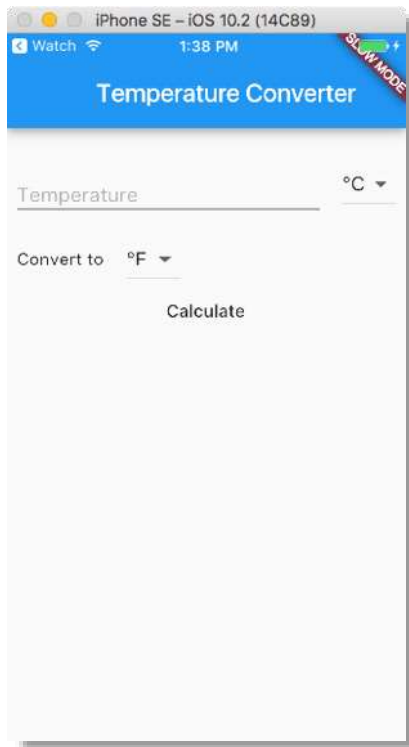
- ▼ Client- and serverside apps
- ▼ Web Apps
 - ▼ Using `dart:html`
 - ▼ [AngularDart](#) – Angular 4 for Dart
 - ▼ [Polymer Dart](#) – Dart-Apps using Polymer
- ▼ Apps für Android und iOS with [Flutter](#)

Web App using `dart:html`

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

Mobile App using Flutter

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



- ▼ Java-, C# and C/C++ developers find their way through easily
- ▼ Terse language
- ▼ Tries to stand in the way as seldom as possible

Thank you very much for listening!

thomas.kuenneth@mathema.de

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

Blog: <http://kuenneth.t.blogspot.de/>