Hannes Koksch | Max Tellmann | Dennis Schmidt

</>

Faszination Flutter

Entwicklung von Web-Anwendungen

Hannes

CSM 2. Semester

~ 2 Jahre Flutter-Erfahrung

Max

CSM 2. Semester

~ Flutter-Neuling

Dennis

CSM 2. Semester

~ 3 Jahre Flutter-Erfahrung

Programm für den ersten Block

1. Theorie

1. Was ist Dart & Flutter?
2. Wie funktioniert Dart & Flutter?
3. Syntax von Dart
4. Was sind Widgets?

2. Praxis

1. Troubleshooting Installation
2. Hello World
3. Training mit Widgets

Was ist Dart & Flutter?

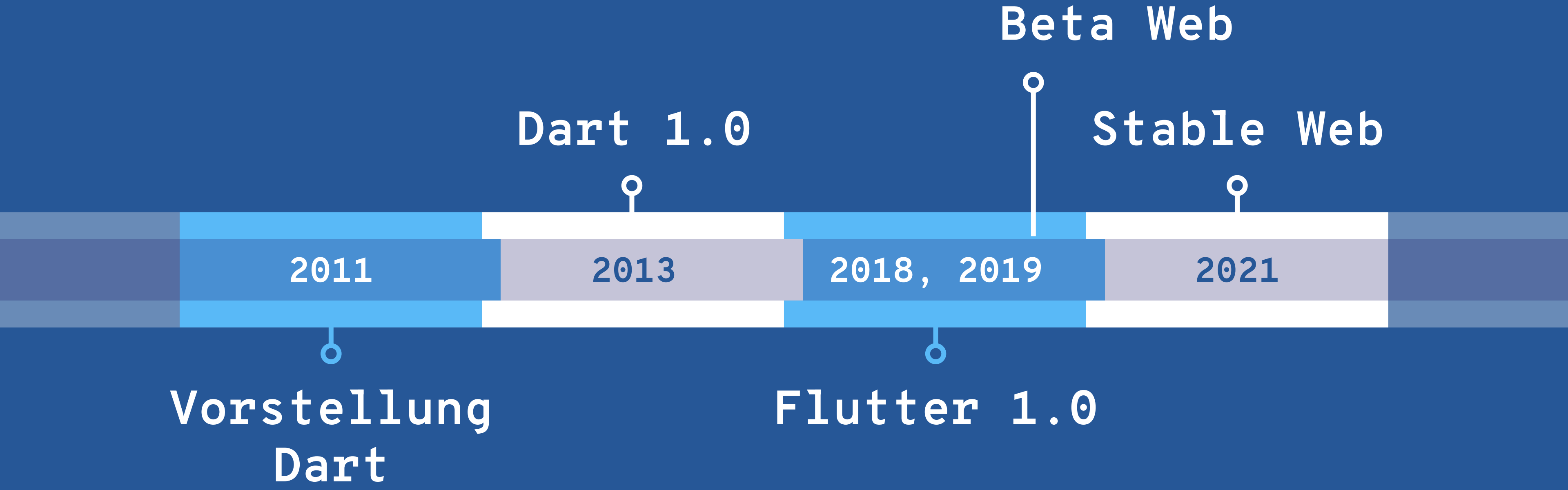
Open-Source Projekt

Cross Platform

Typisierte Sprache

Schnelle Entwicklung

Geschichte von Dart & Flutter



Wie funktioniert Dart & Flutter?

Build Mode

Compiling

Rendering

https://github.com/flutter/flutter/wiki/Flutter%27s-modes

Just In Time (JIT) vs. Ahead Of Time (AOT)

Just In Time (JIT)

- langsamer Start (warm up)
- Build ist schneller als bei AOT
- Code wird zur Runtime kompiliert
- Debugging-Suite
- Hot Reload / Hot Restart
- für das Development

```
dart run [file path]
```

Ahead Of Time (AOT)

- schneller Start
- Code ist vor Runtime kompiliert
- keine Debugging-Tools
- echte Performance
- für Production

```
dart compile web [file path]
```

Wie funktioniert Dart & Flutter?

Build Mode

Compiling

Rendering

https://github.com/flutter/flutter/wiki/Flutter%27s-modes

Render engines (web)

HTML-Renderer

- Kleinere Downloadgröße
- Verwendet Kombination aus HTML, CSS, Canvas-Elemente und SVGs
- Darstellungen bei komplizierten Layouts, z. B. mit Schatten, nicht immer wie gewollt
- Weniger CORS-Probleme bei Images

CanvasKit

- Vollständig konsistent mit Flutter Mobile und Desktop
- Bessere Performance
- Verwendet WebGL zum Rendern von Skia-Malbefehlen.
- Zusätzlich 1.5 mb Download

https://docs.flutter.dev/platform-integration/web/renderers

Render Engines in Flutter Web

Verwendung

```
flutter build web --web-renderer canvaskit
flutter build web --web-renderer html
flutter build web --web-renderer auto
```

Funktioniert auch mit

```
flutter run web --web-renderer [...]
```

Option `auto`:
(Default) Verwendet canvaskit für Desktop und html für Anfragen von mobile Browsers

https://docs.flutter.dev/platform-integration/web/renderers

Syntax

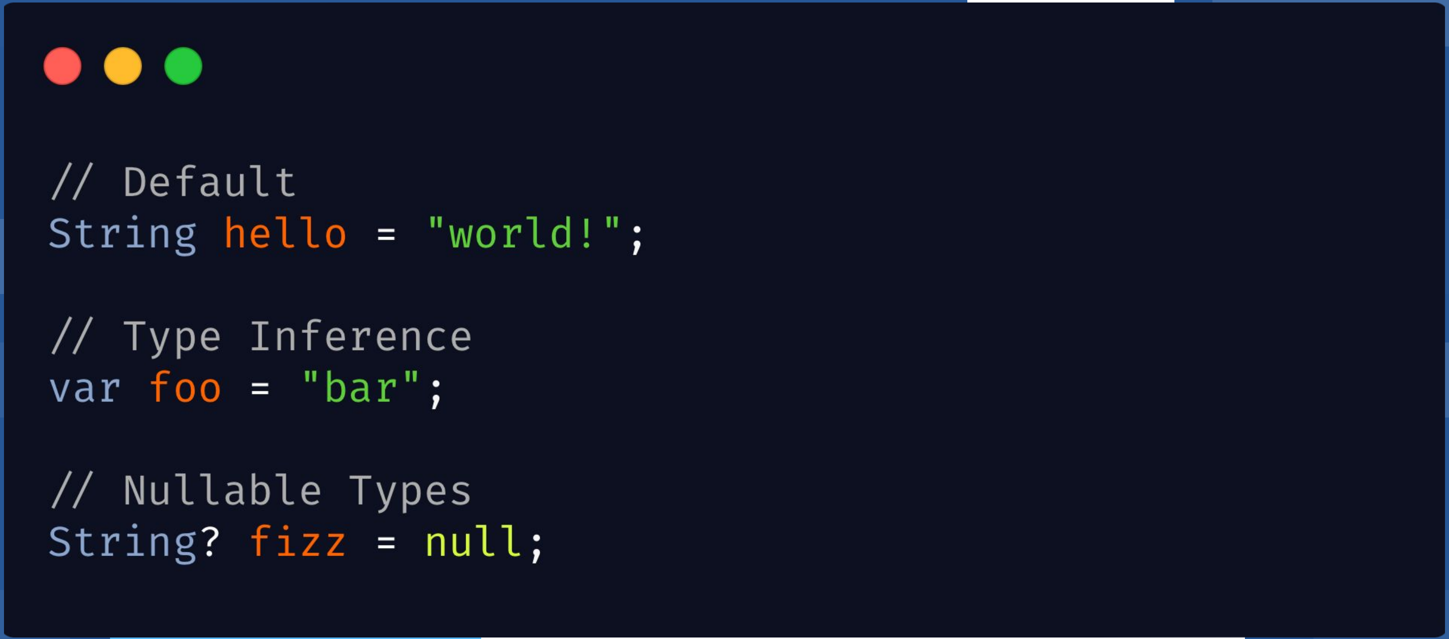
aka. “Wie schreib ich Dart? For Beginners”

Variablen & Datentypen

int, double
String
bool
List<T>
Map<K,V>
Set<T>
null

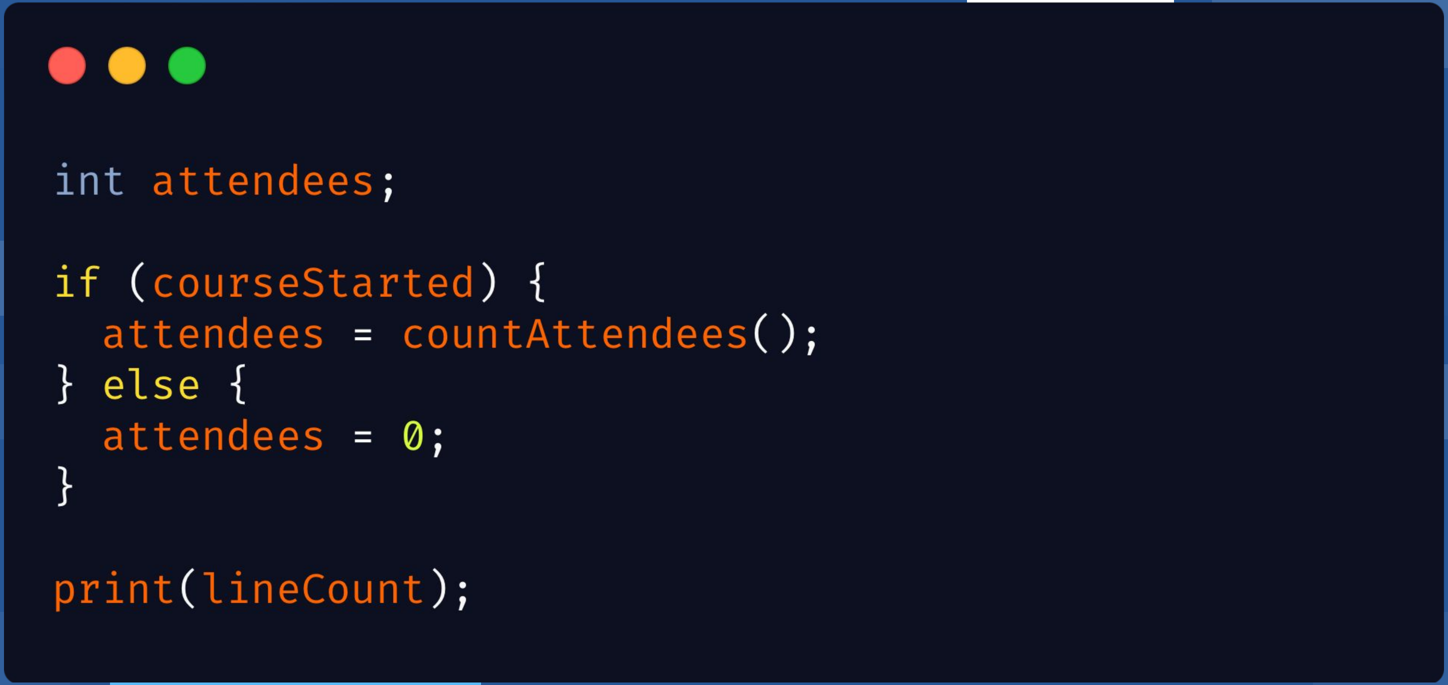
Record
Rune
Symbol

https://dart.dev/language/built-in-types



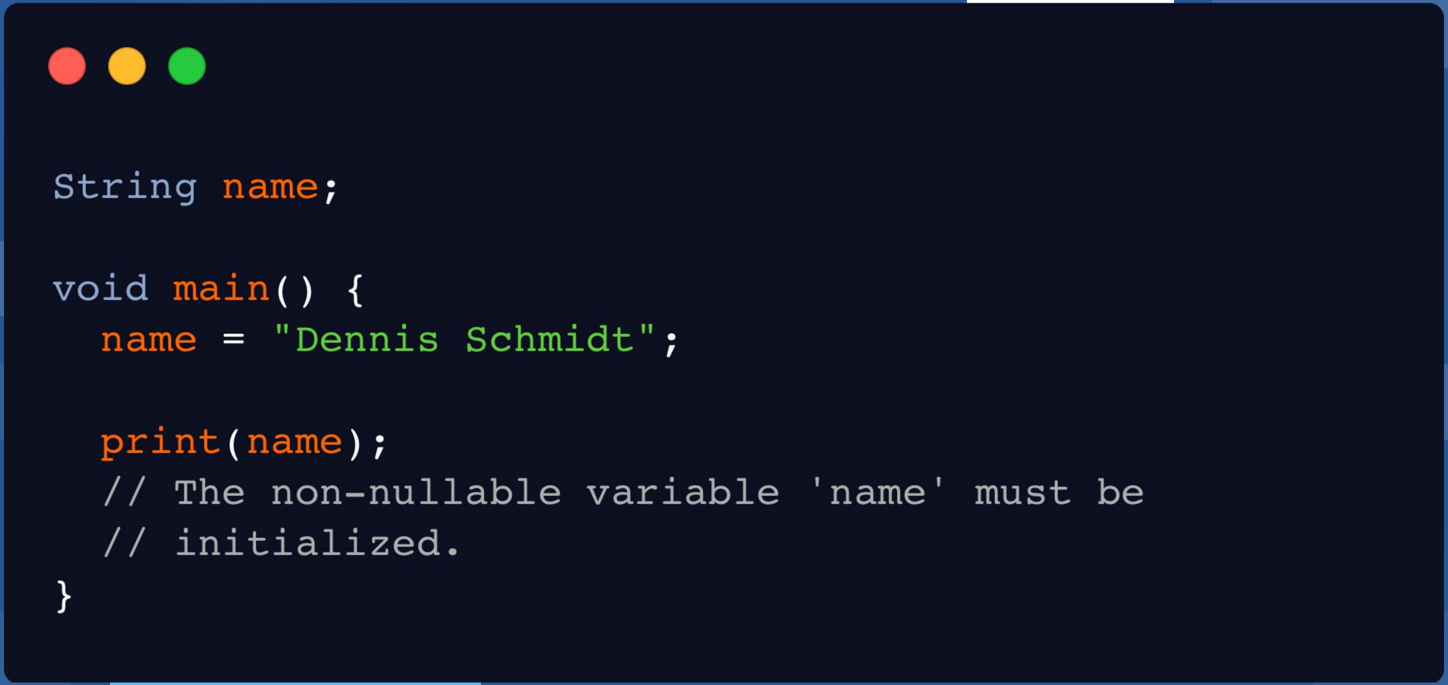
```dart
// Default
String hello = "world!";

// Type Inference
var foo = "bar";

// Nullable Types
String? fizz = null;
```

https://dart.dev/language/variables



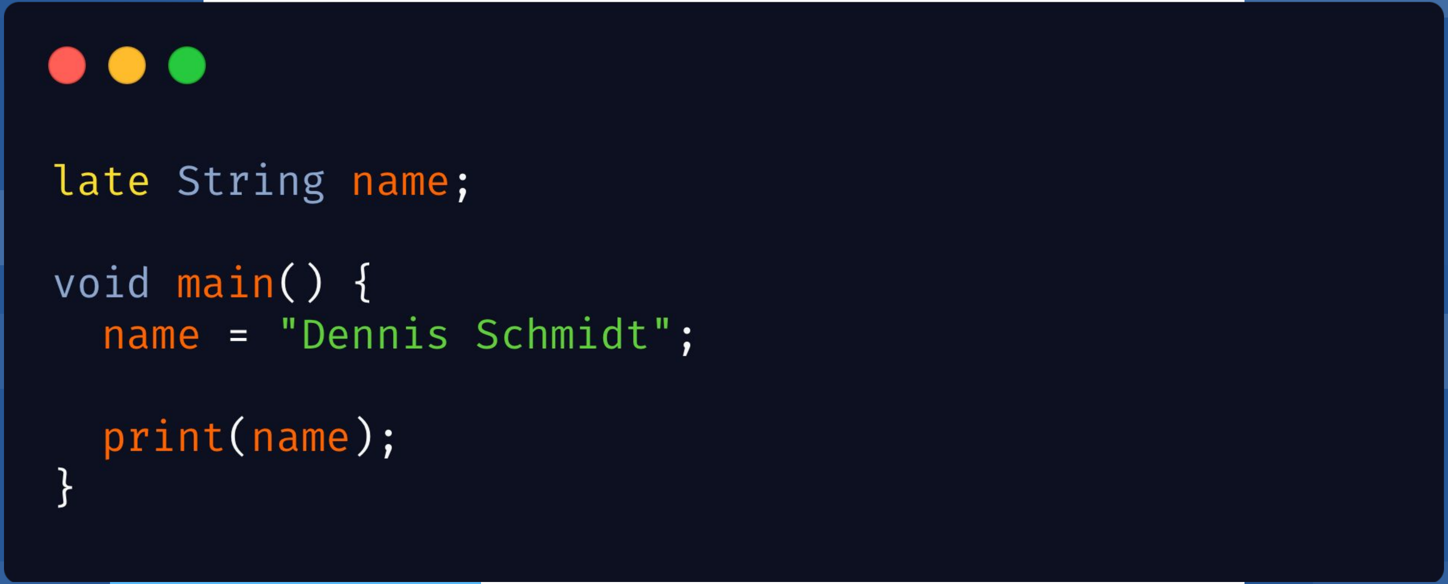
```dart
int attendees;

if (courseStarted) {
  attendees = countAttendees();
} else {
  attendees = 0;
}

print(lineCount);
```

https://dart.dev/language/variables



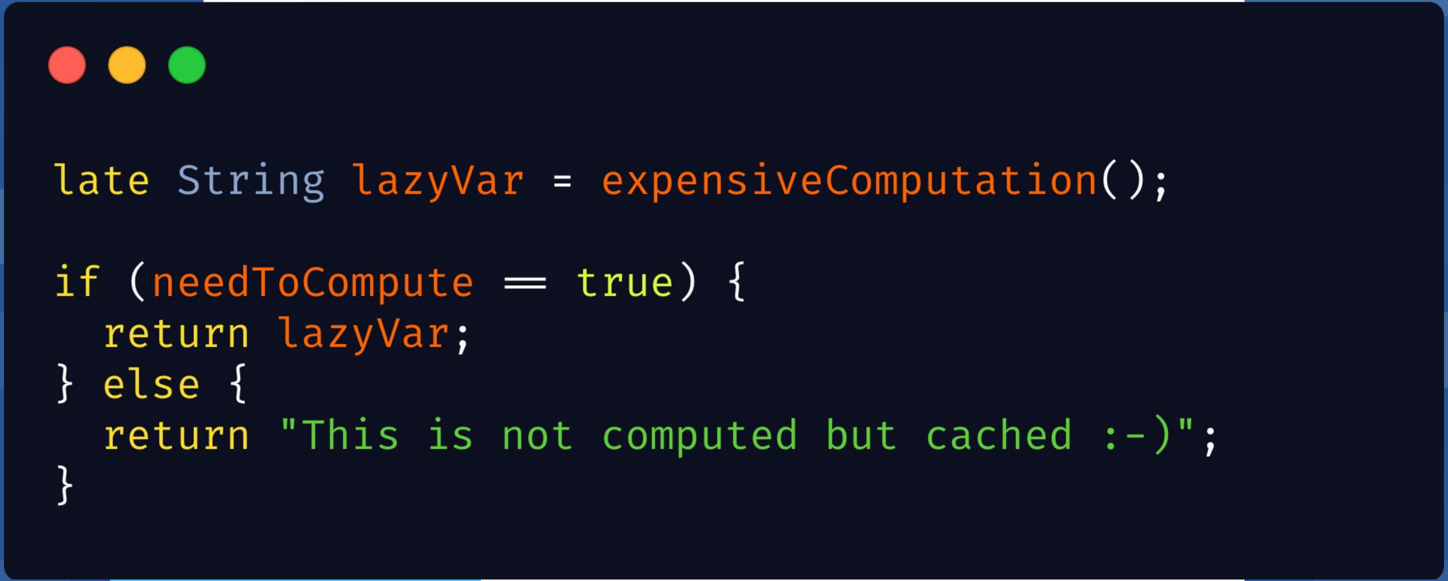
```
String name;

void main() {
  name = "Dennis Schmidt";

  print(name);
  // The non-nullable variable 'name' must be
  // initialized.
}
```

https://dart.dev/language/variables



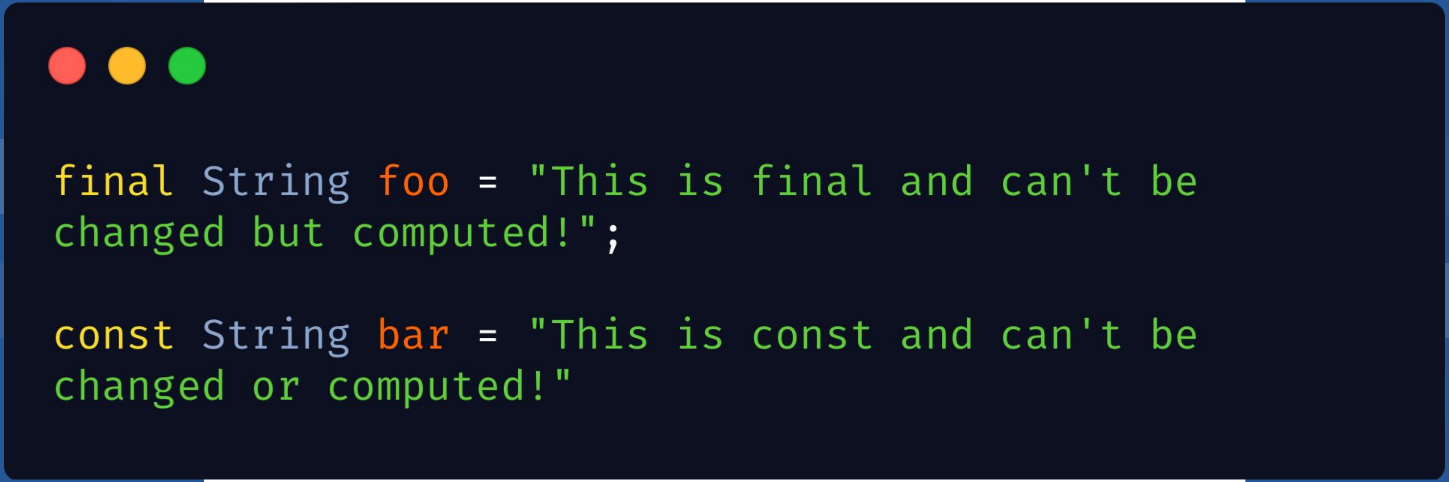
```dart
late String name;

void main() {
  name = "Dennis Schmidt";

  print(name);
}
```

https://dart.dev/language/variables



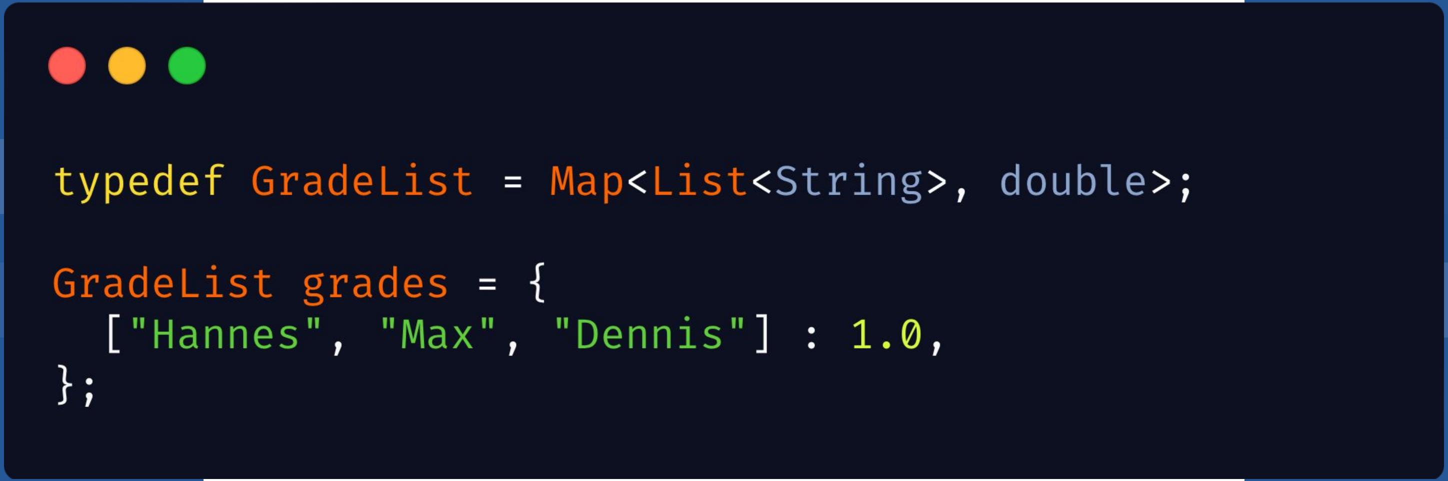
```dart
late String lazyVar = expensiveComputation();

if (needToCompute == true) {
  return lazyVar;
} else {
  return "This is not computed but cached :-)";
}
```

https://dart.dev/language/variables



```dart
final String foo = "This is final and can't be
changed but computed!";

const String bar = "This is const and can't be
changed or computed!"
```

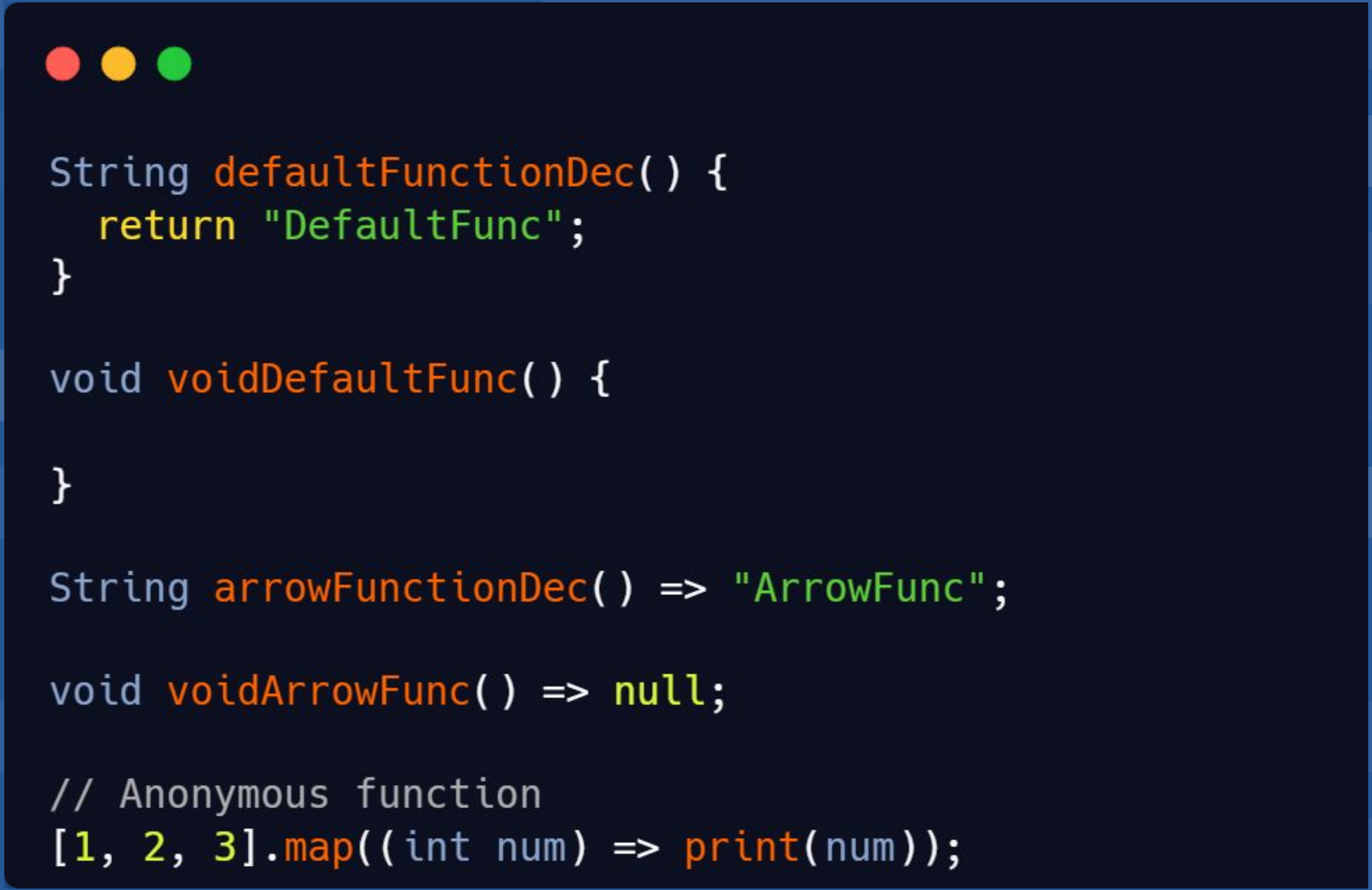
https://dart.dev/language/variables



```
typedef GradeList = Map<List<String>, double>;

GradeList grades = {
  ["Hannes", "Max", "Dennis"] : 1.0,
};
```

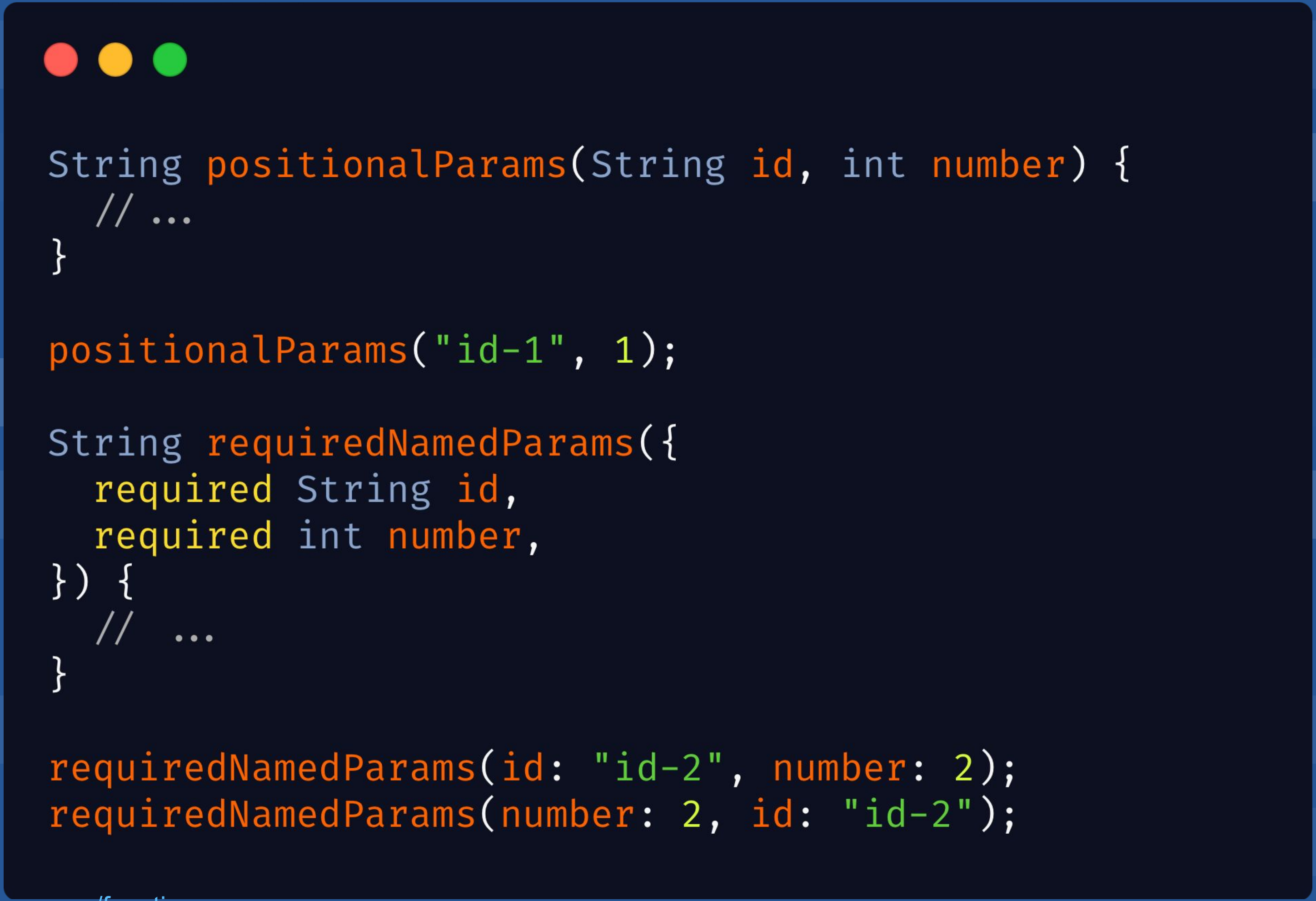
https://dart.dev/language/typedefs

Funktionen



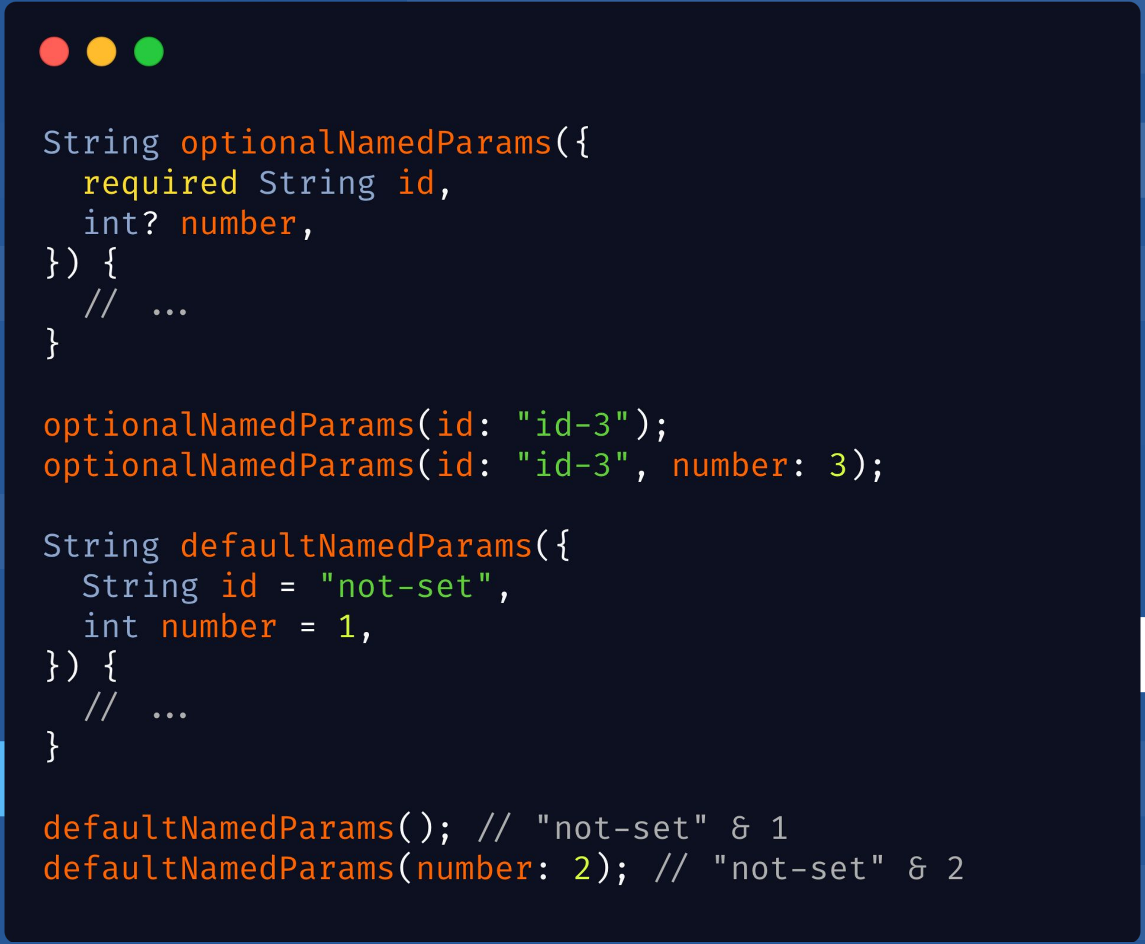
```dart
String defaultFunctionDec() {
  return "DefaultFunc";
}

void voidDefaultFunc() {

}

String arrowFunctionDec() => "ArrowFunc";

void voidArrowFunc() => null;

// Anonymous function
[1, 2, 3].map((int num) => print(num));
```

https://dart.dev/language/functions



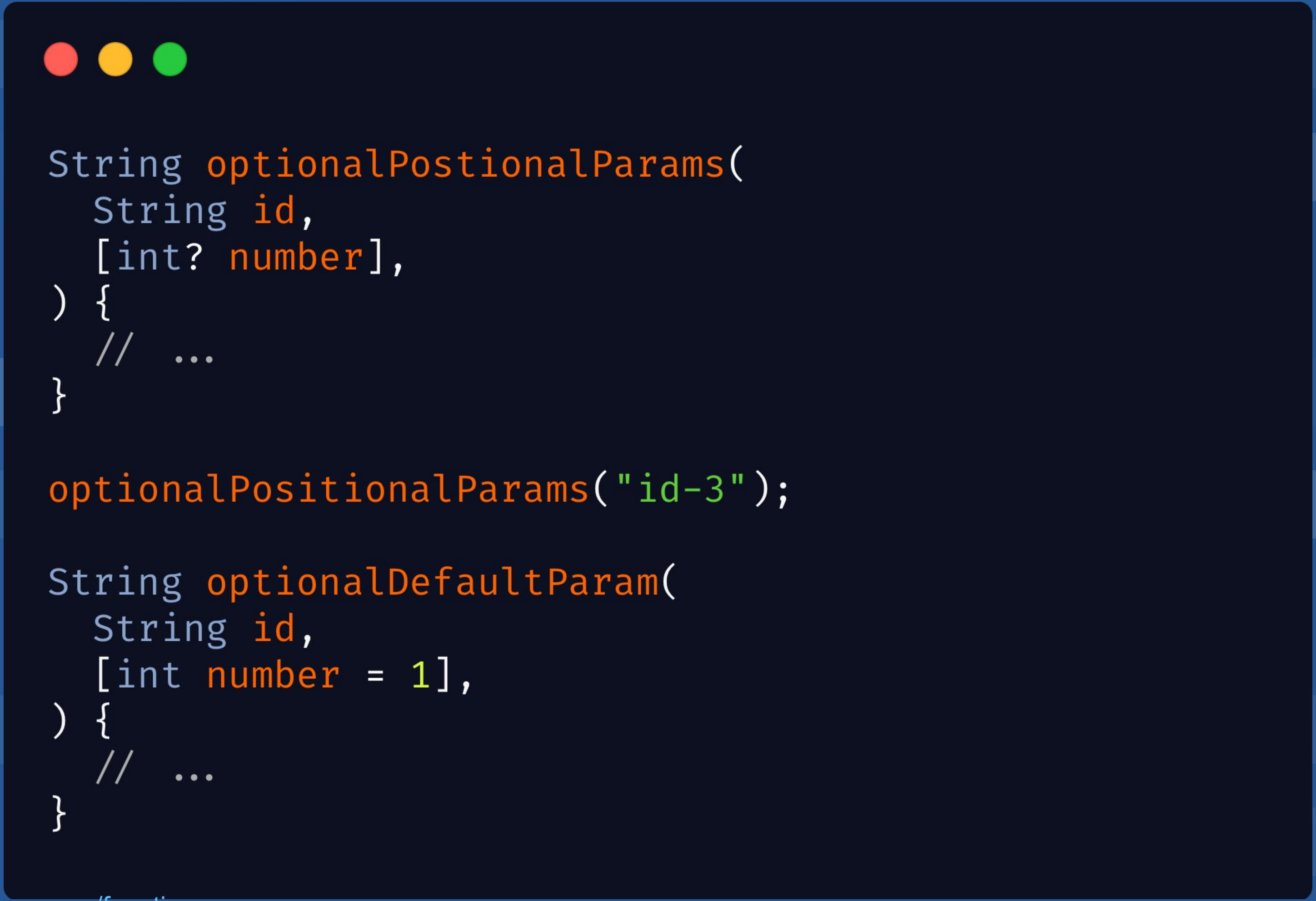
```dart
String positionalParams(String id, int number) {
  // ...
}

positionalParams("id-1", 1);

String requiredNamedParams({
  required String id,
  required int number,
}) {
  //  ...
}

requiredNamedParams(id: "id-2", number: 2);
requiredNamedParams(number: 2, id: "id-2");
```

https://dart.dev/language/functions



```dart
String optionalNamedParams({
  required String id,
  int? number,
}) {
  // ...
}

optionalNamedParams(id: "id-3");
optionalNamedParams(id: "id-3", number: 3);

String defaultNamedParams({
  String id = "not-set",
  int number = 1,
}) {
  // ...
}

defaultNamedParams(); // "not-set" & 1
defaultNamedParams(number: 2); // "not-set" & 2
```

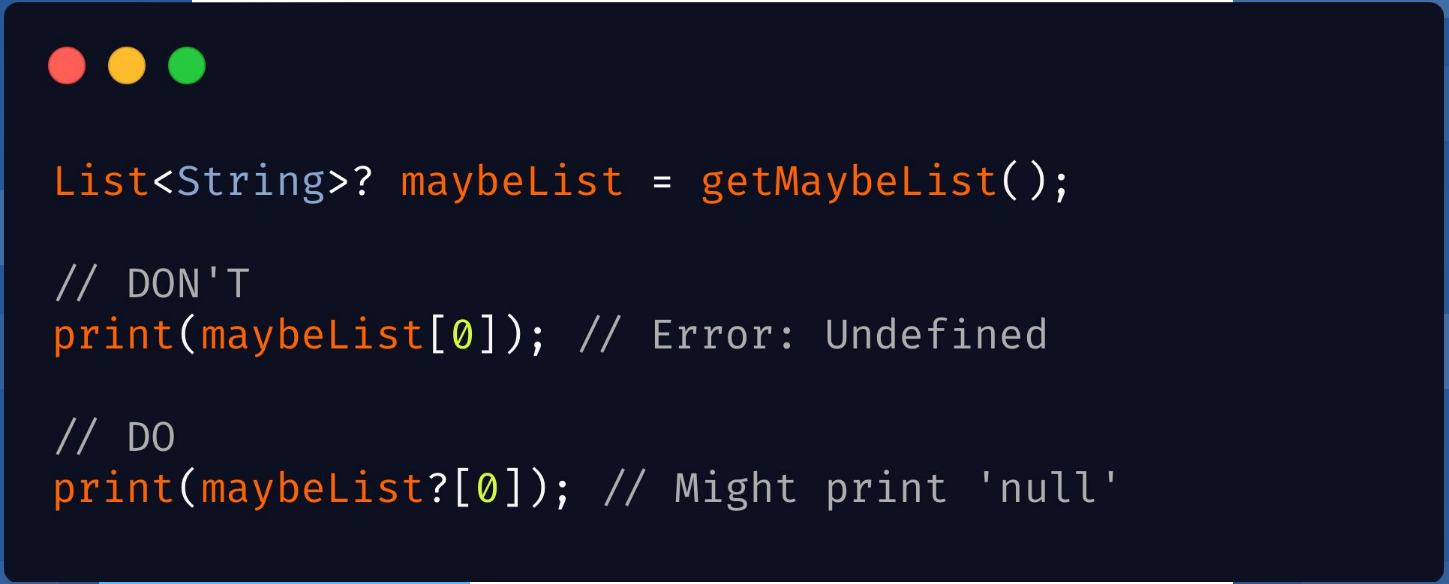
https://dart.dev/language/functions



```
String optionalPostionalParams(
  String id,
  [int? number],
) {
  // ...
}

optionalPositionalParams("id-3");

String optionalDefaultParam(
  String id,
  [int number = 1],
) {
  // ...
}
```

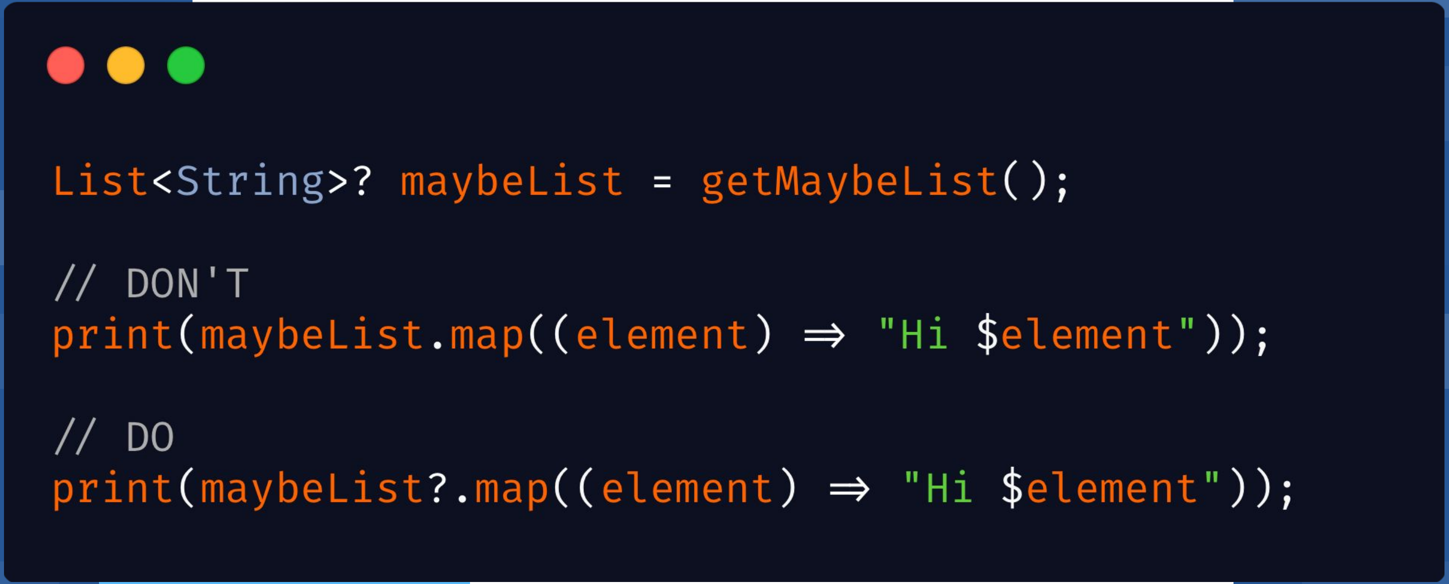
https://dart.dev/language/functions

Operationen



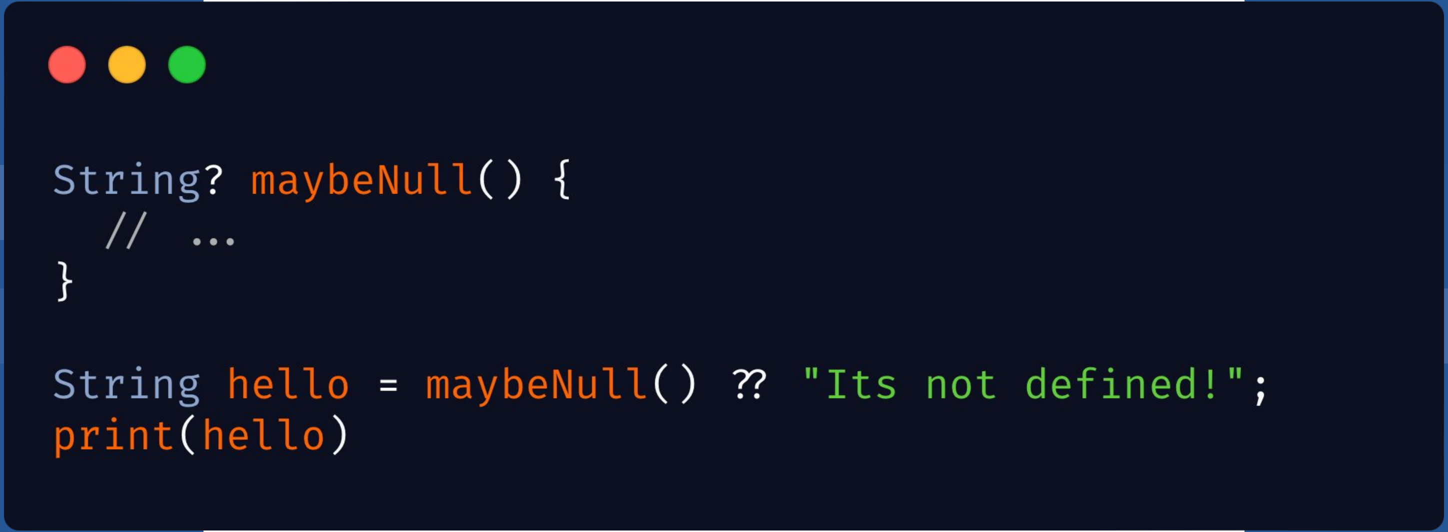
```dart
List<String>? maybeList = getMaybeList();

// DON'T
print(maybeList[0]); // Error: Undefined

// DO
print(maybeList?[0]); // Might print 'null'
```

https://dart.dev/language/operators



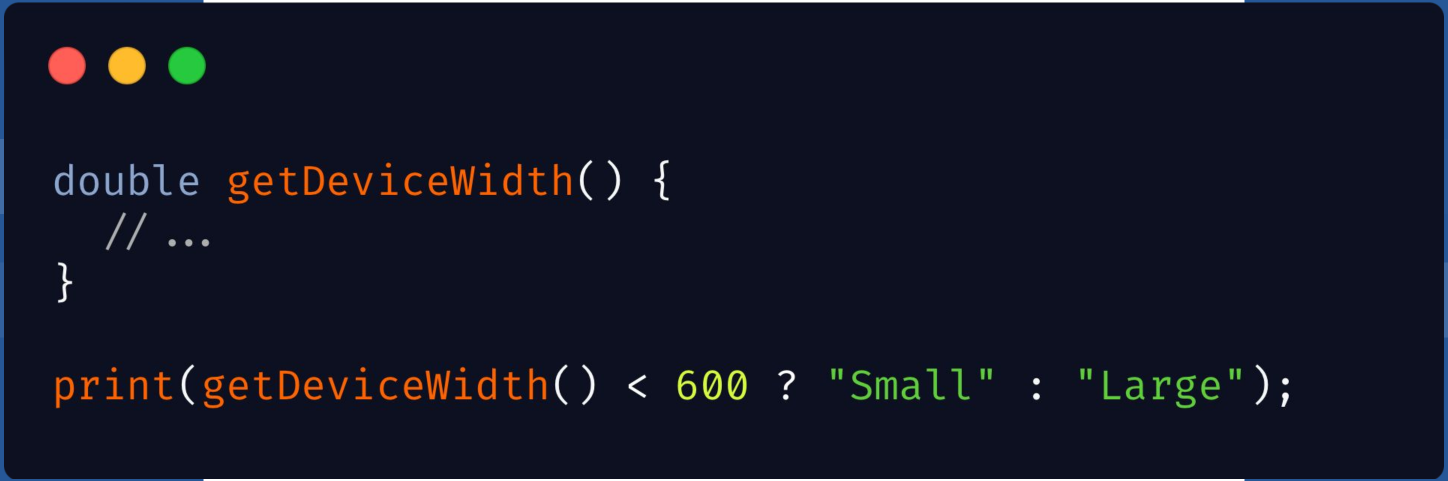
```dart
List<String>? maybeList = getMaybeList();

// DON'T
print(maybeList.map((element) => "Hi $element"));

// DO
print(maybeList?.map((element) => "Hi $element"));
```

https://dart.dev/language/operators



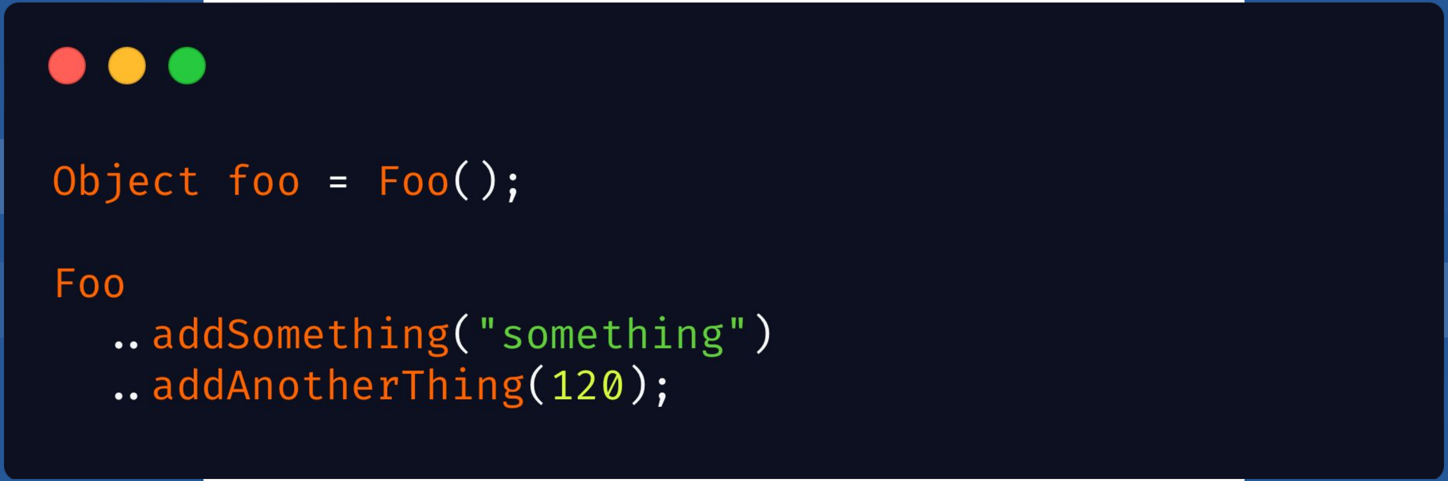
```
String? maybeNull() {
  // ...
}

String hello = maybeNull() ?? "Its not defined!";
print(hello)
```

https://dart.dev/language/operators



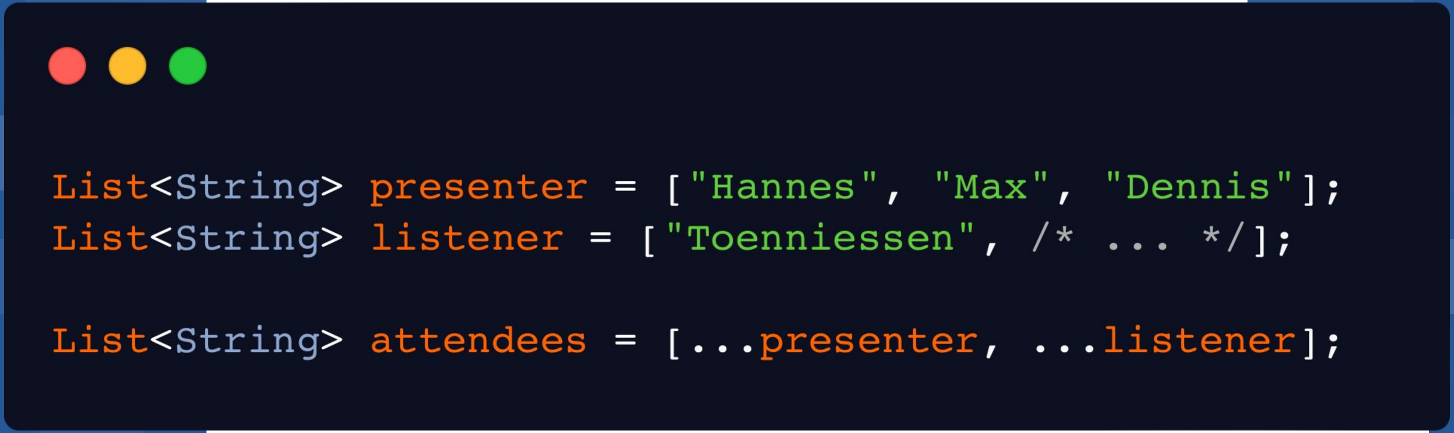
```dart
double getDeviceWidth() {
  // ...
}

print(getDeviceWidth() < 600 ? "Small" : "Large");
```

https://dart.dev/language/operators



```
Object foo = Foo();

Foo
  ..addSomething("something")
  ..addAnotherThing(120);
```

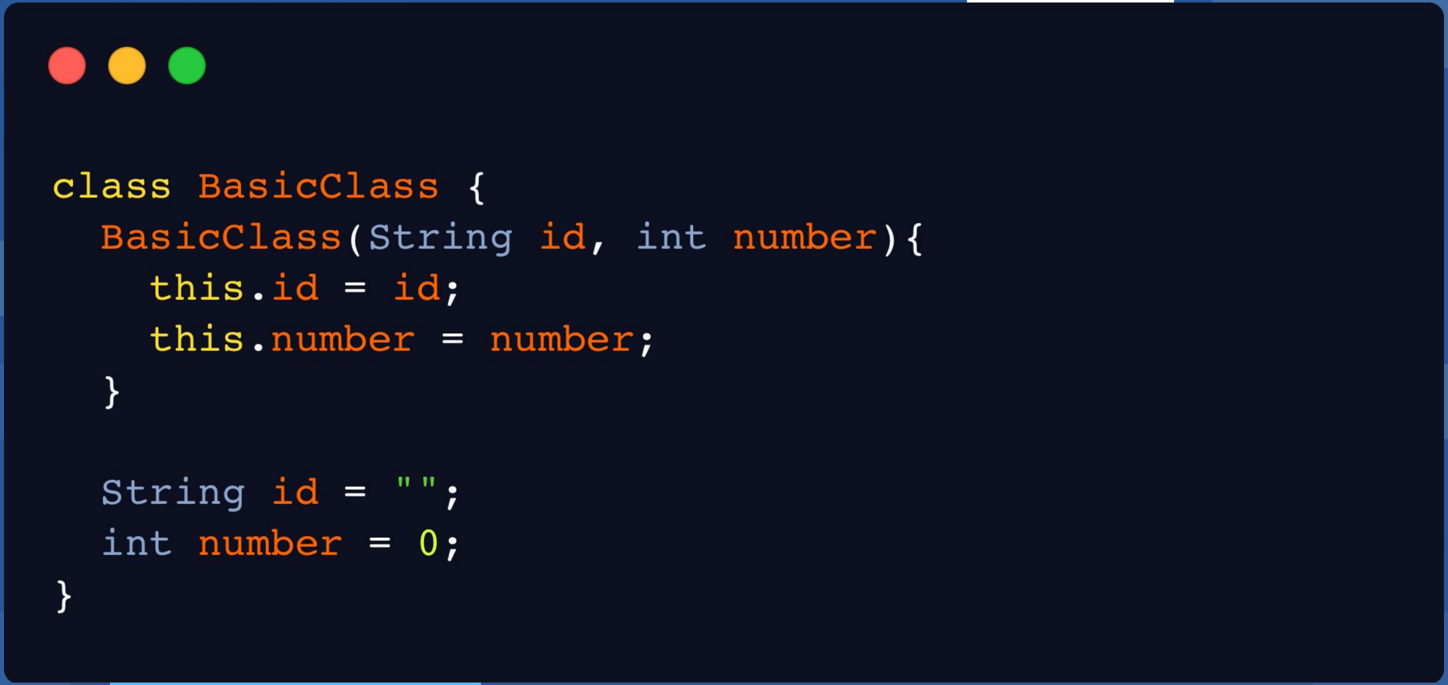
https://dart.dev/language/operators



```dart
List<String> presenter = ["Hannes", "Max", "Dennis"];
List<String> listener = ["Toenniessen", /* ... */];

List<String> attendees = [...presenter, ...listener];
```

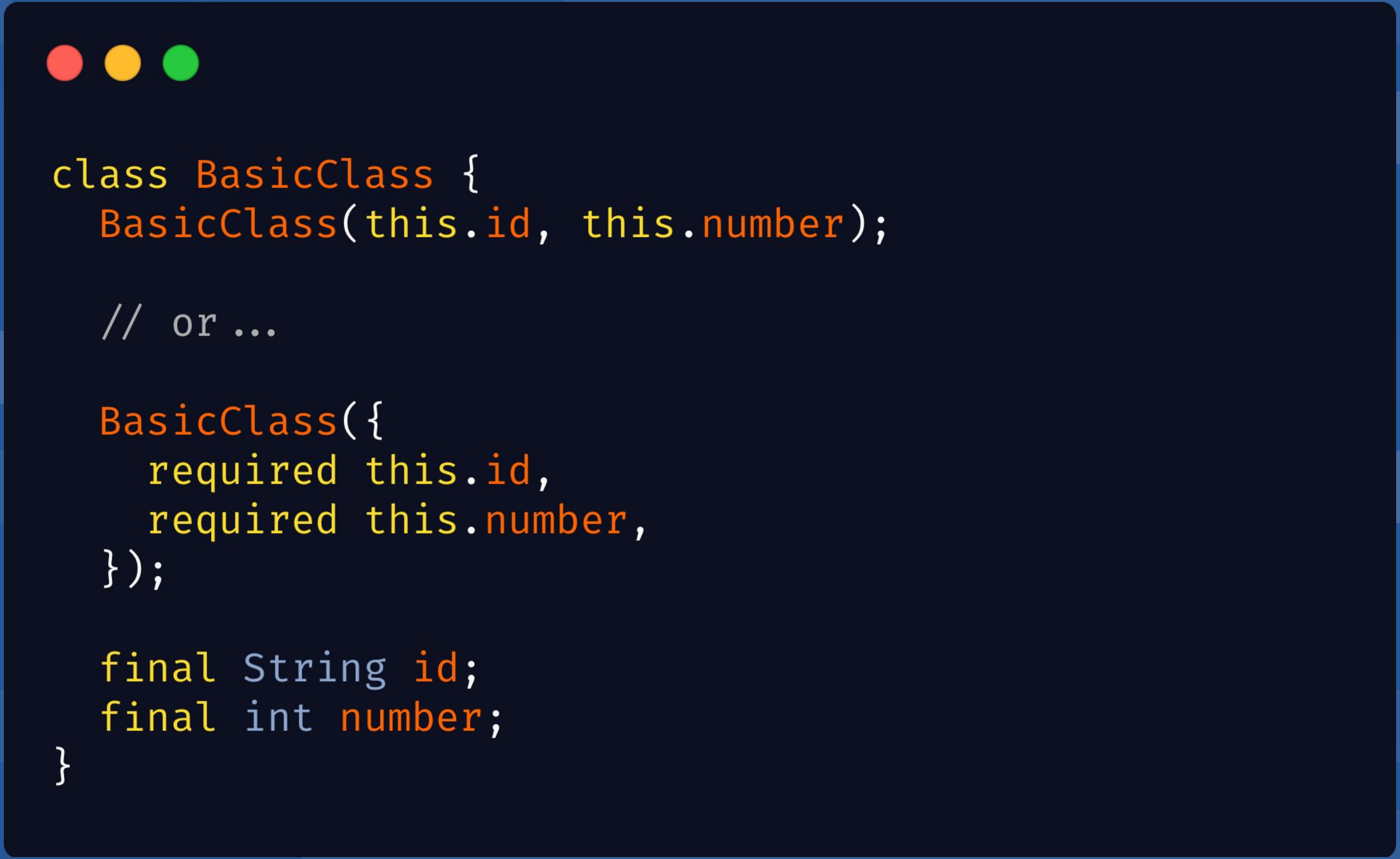
https://dart.dev/language/operators

Klassen



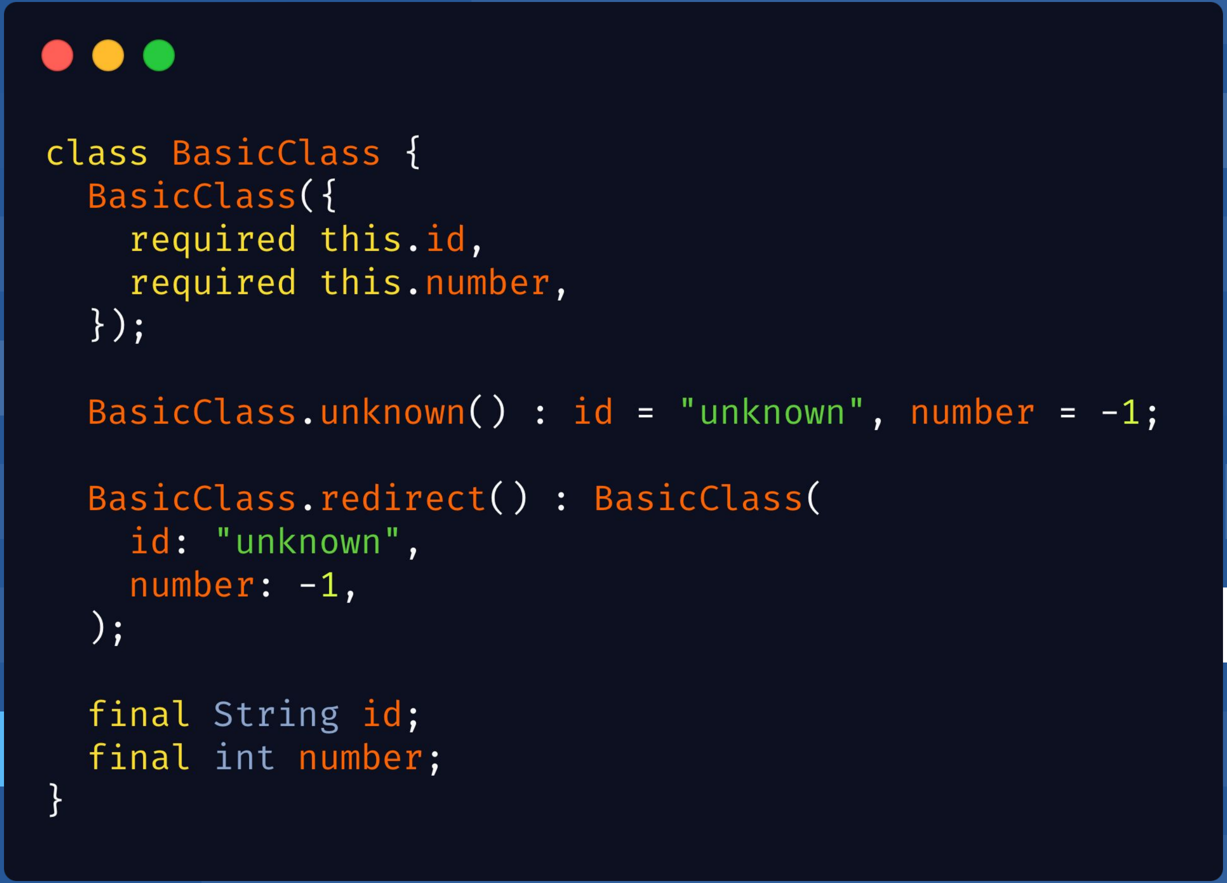
```
class BasicClass {
  BasicClass(String id, int number){
    this.id = id;
    this.number = number;
  }

  String id = "";
  int number = 0;
}
```

https://dart.dev/language/classes



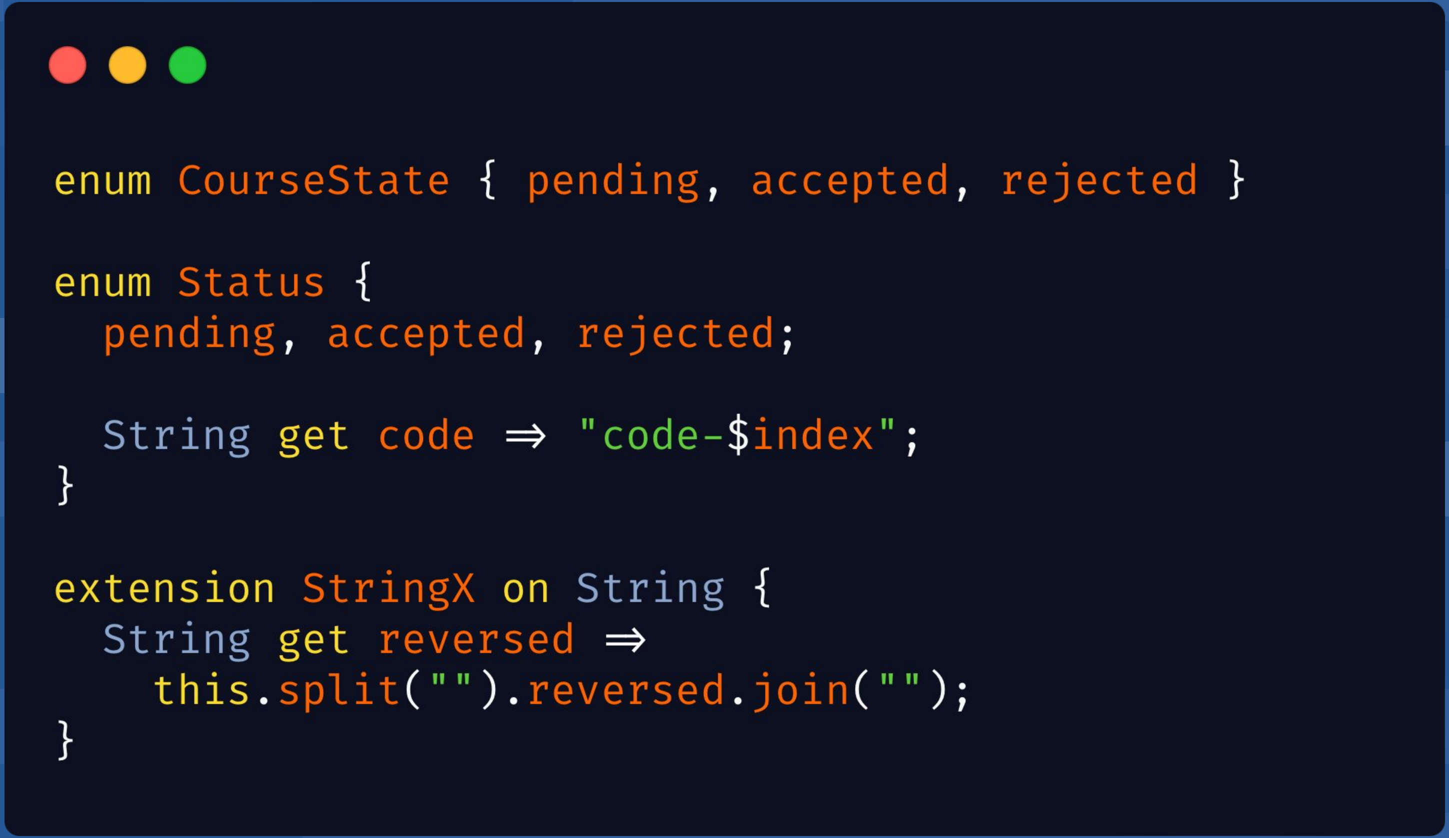
```
class BasicClass {
  BasicClass(this.id, this.number);

  // or...

  BasicClass({
    required this.id,
    required this.number,
  });

  final String id;
  final int number;
}
```

https://dart.dev/language/classes



```
class BasicClass {
  BasicClass({
    required this.id,
    required this.number,
  });

  BasicClass.unknown() : id = "unknown", number = -1;

  BasicClass.redirect() : BasicClass(
    id: "unknown",
    number: -1,
  );

  final String id;
  final int number;
}
```

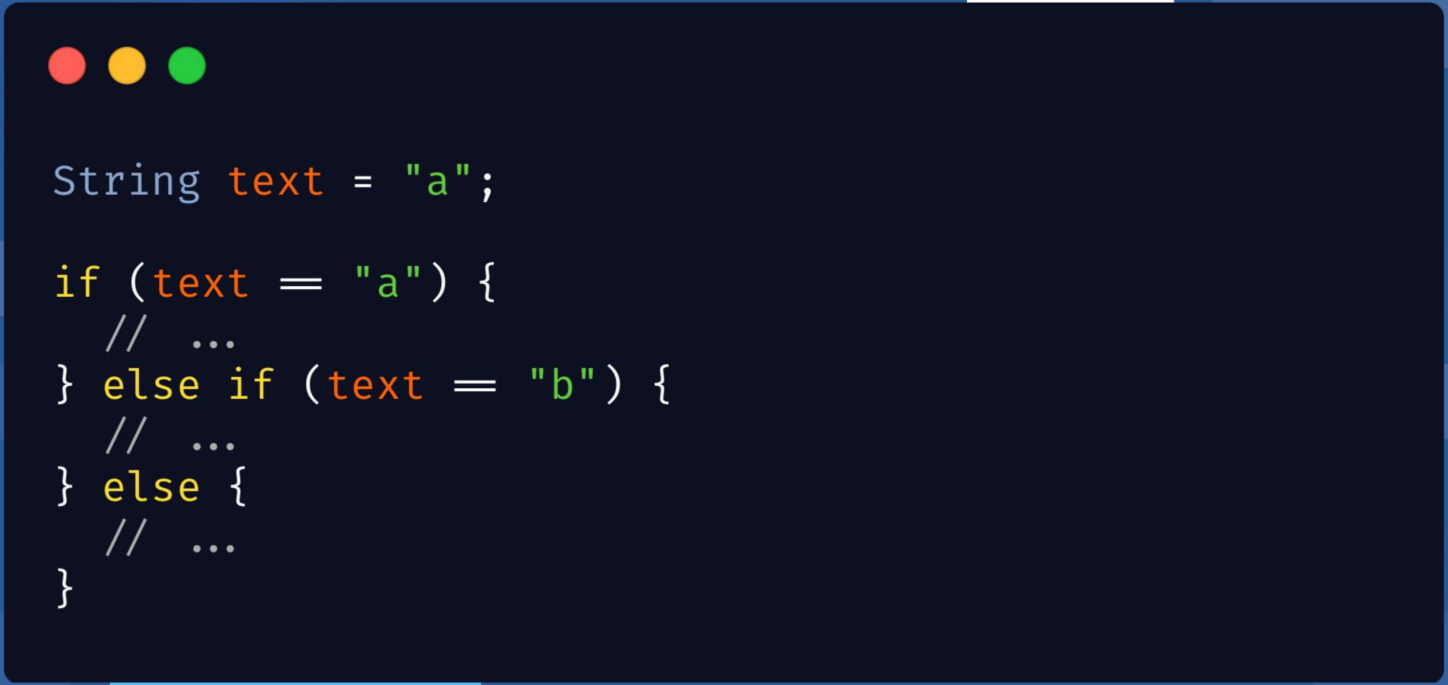
https://dart.dev/language/classes



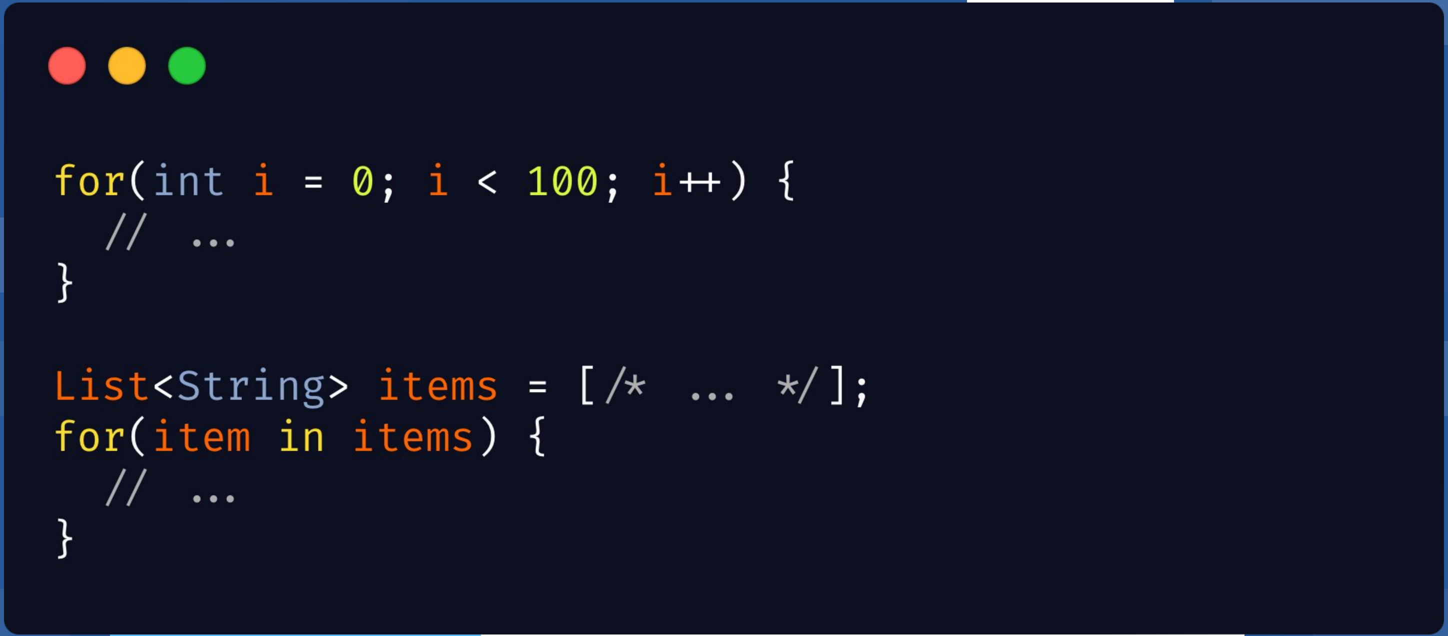
```
enum CourseState { pending, accepted, rejected }

enum Status {
  pending, accepted, rejected;

  String get code => "code-$index";
}

extension StringX on String {
  String get reversed =>
    this.split("").reversed.join("");
}
```

https://dart.dev/language/classes

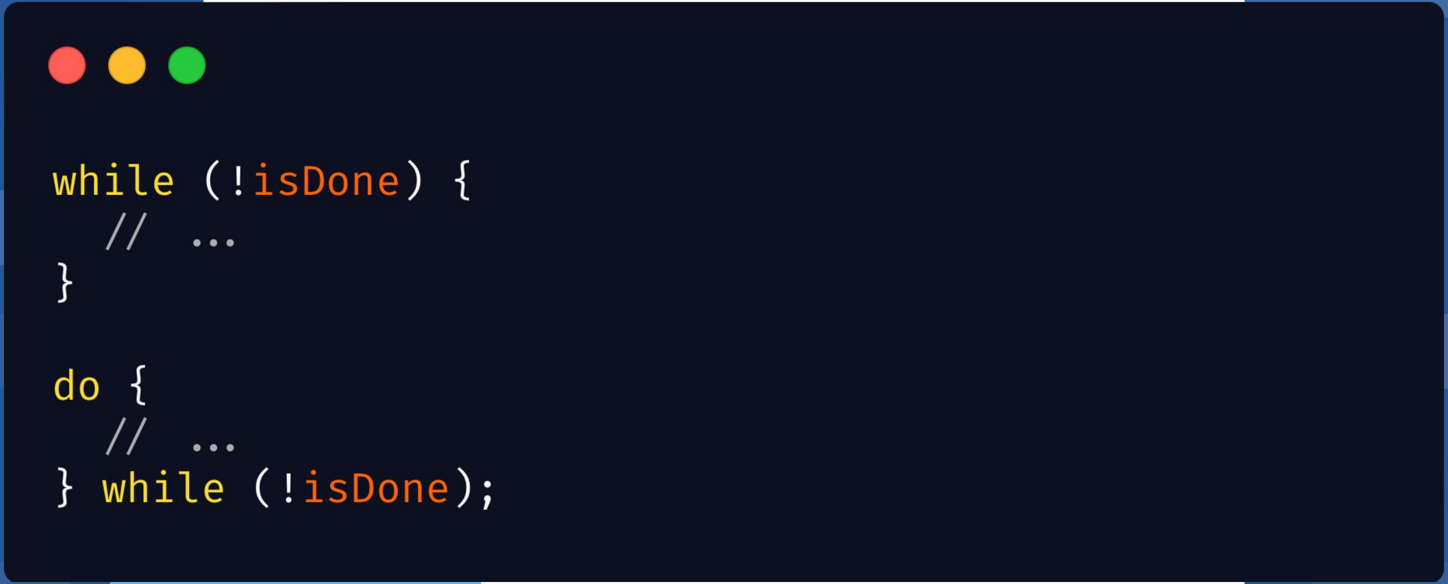
Kontrollstrukturen

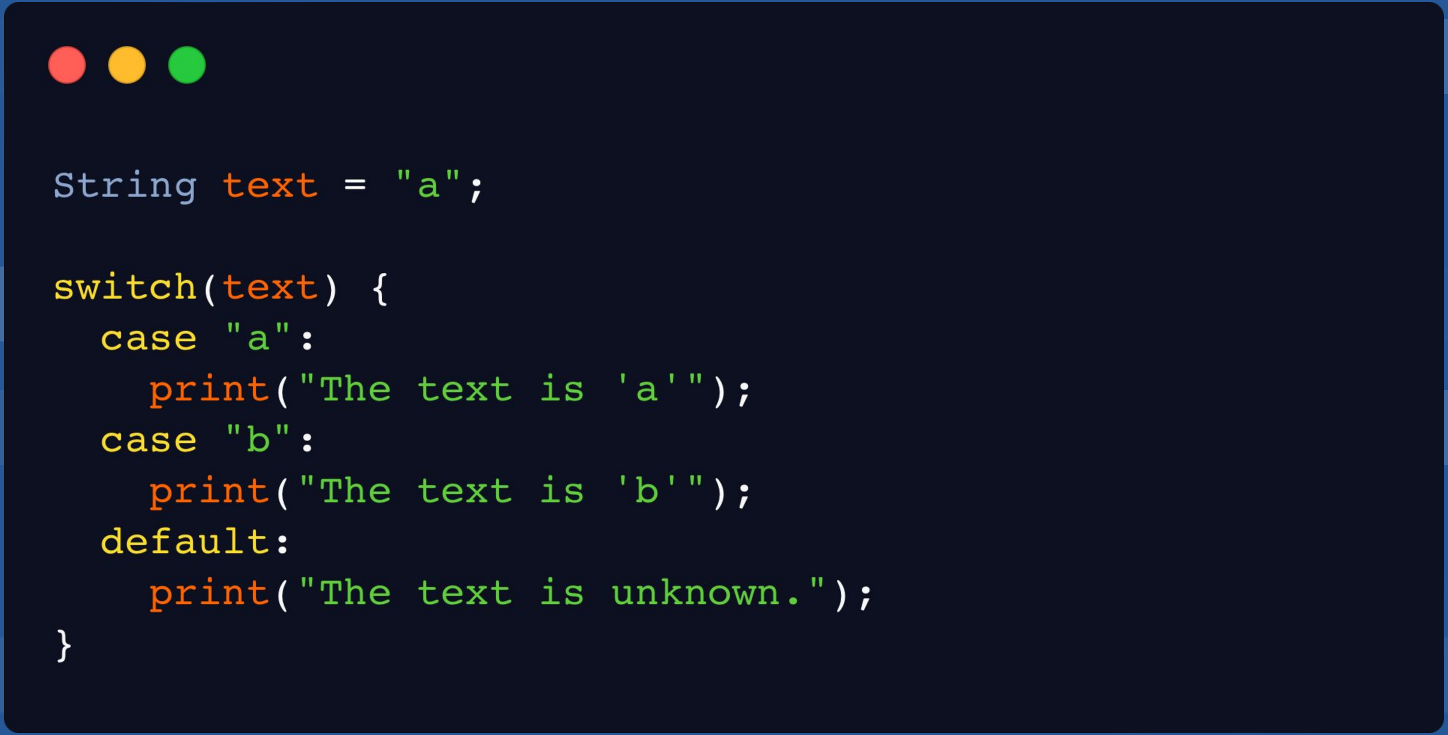


```java
String text = "a";

if (text == "a") {
  // ...
} else if (text == "b") {
  // ...
} else {
  // ...
}
```

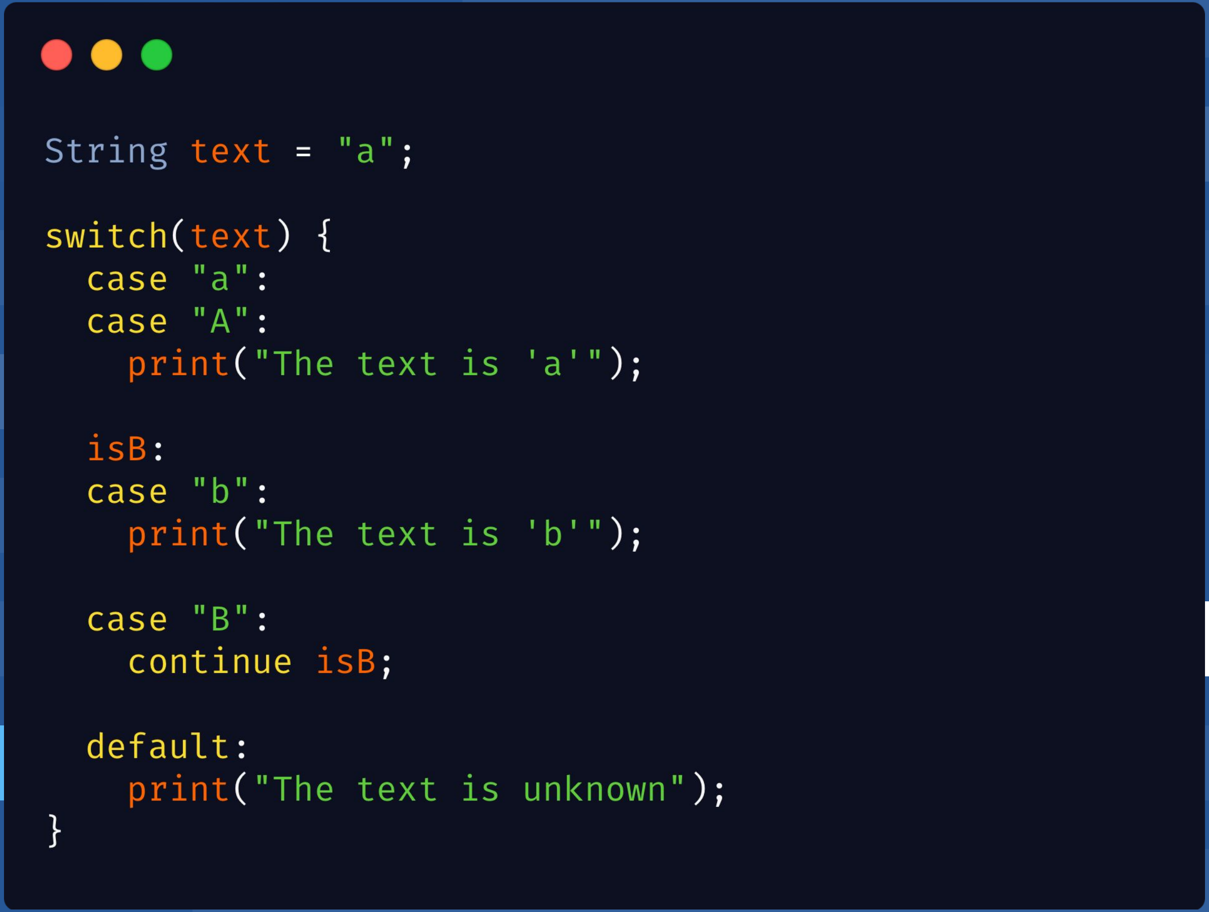


```
for(int i = 0; i < 100; i++) {
  // ...
}

List<String> items = [/* ... */];
for(item in items) {
  // ...
}
```

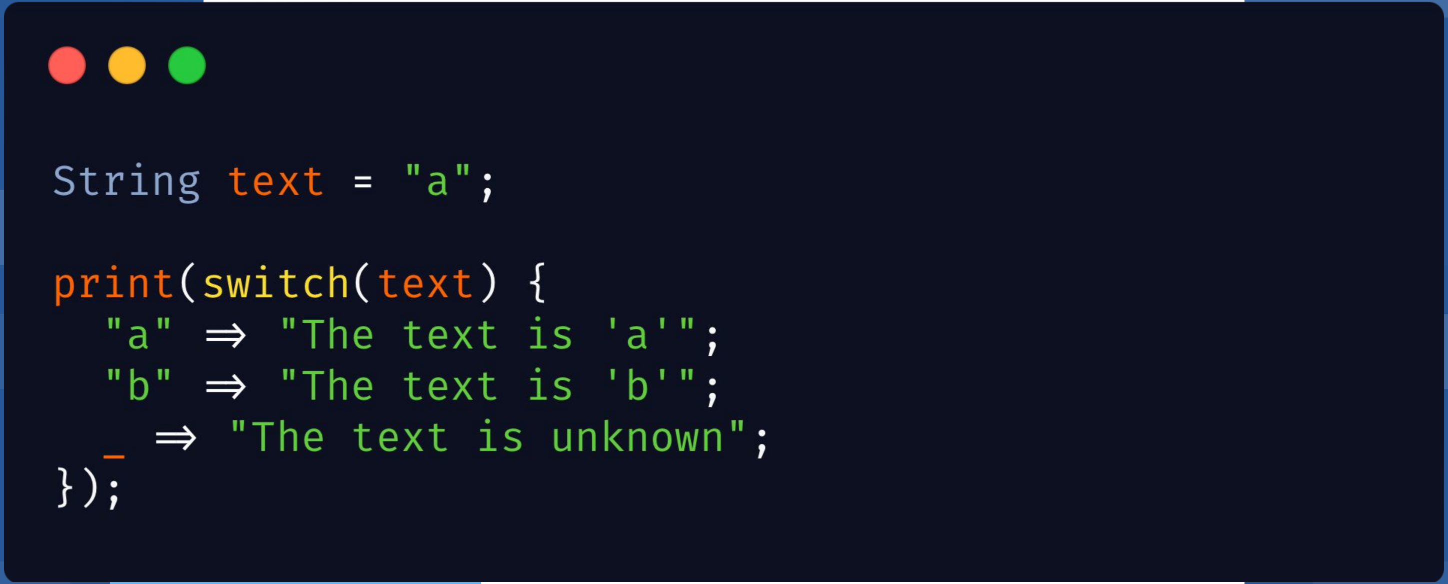

```
while (!isDone) {
  // ...
}

do {
  // ...
} while (!isDone);
```



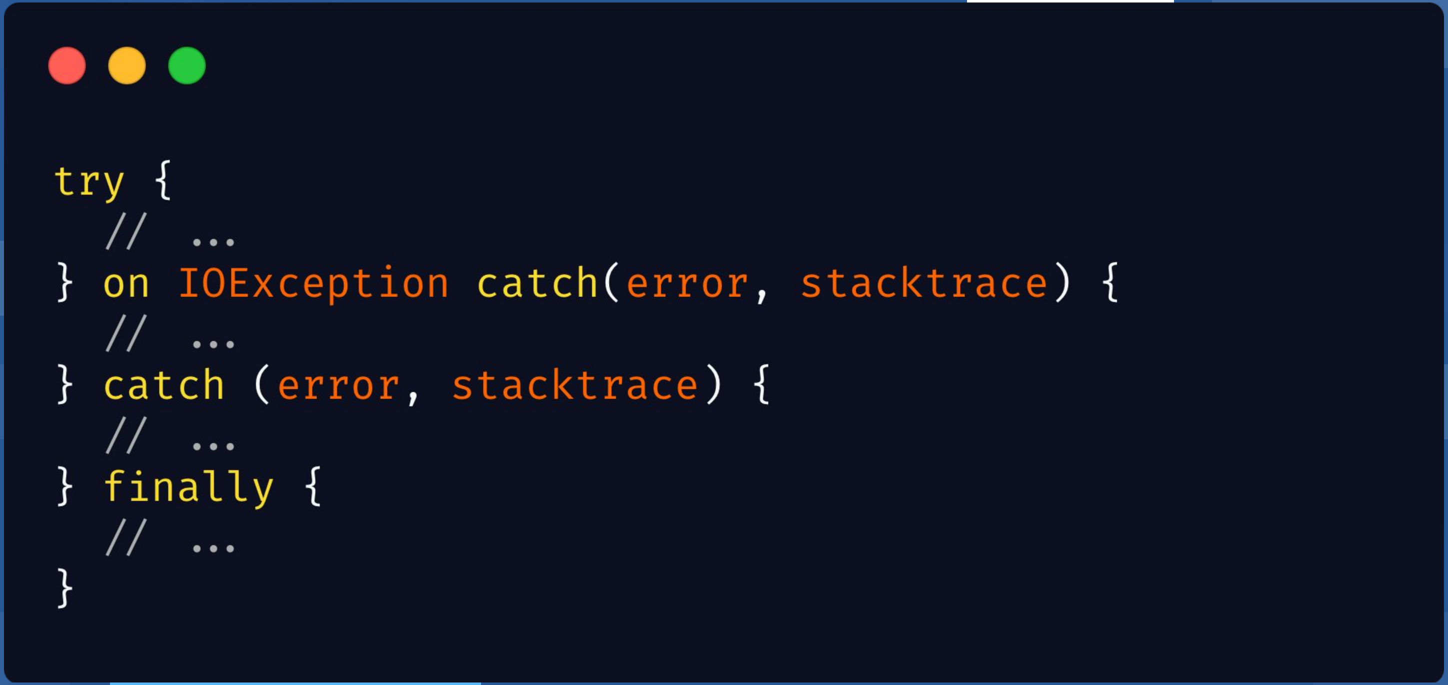
```
String text = "a";

switch(text) {
  case "a":
    print("The text is 'a'");
  case "b":
    print("The text is 'b'");
  default:
    print("The text is unknown.");
}
```



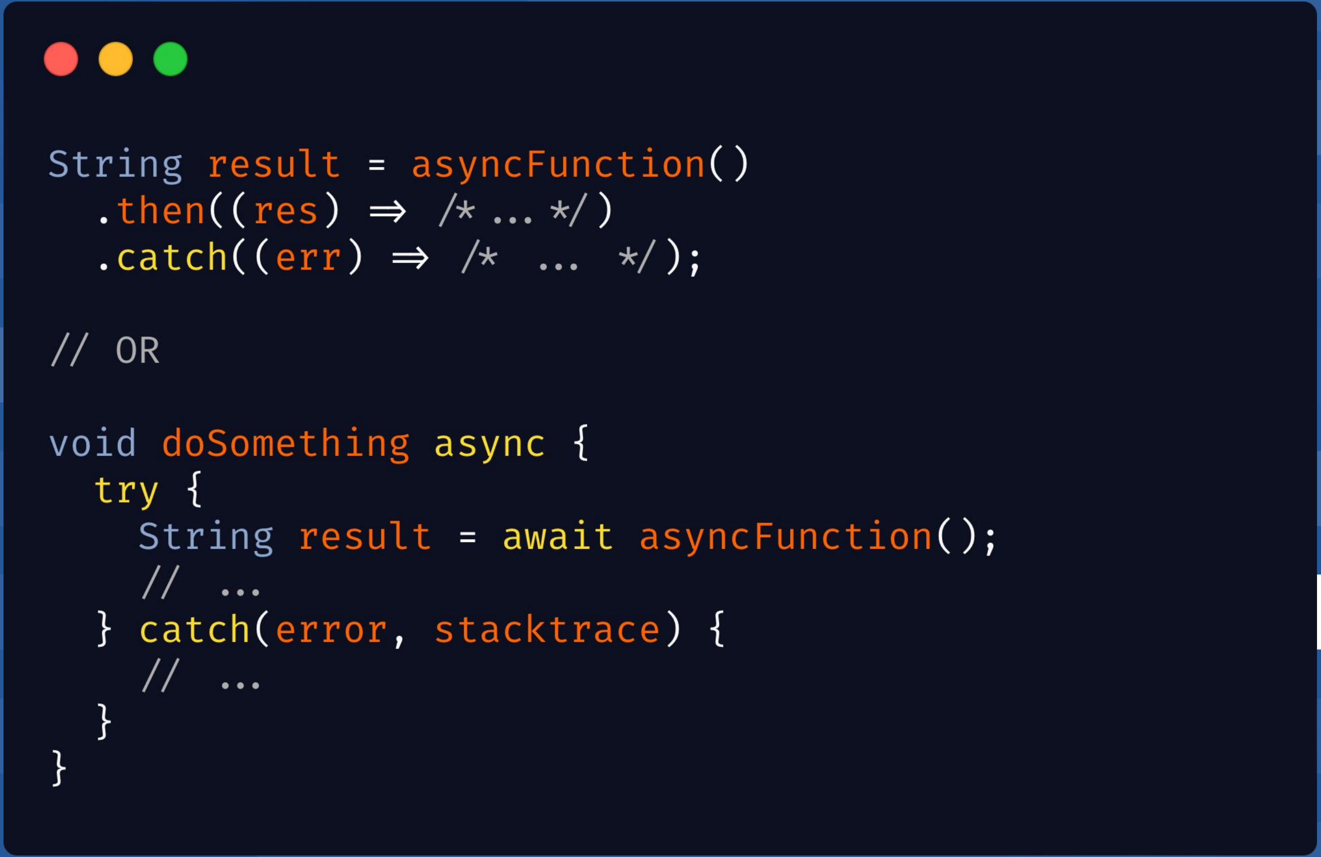
```
String text = "a";

switch(text) {
  case "a":
  case "A":
    print("The text is 'a'");

  isB:
  case "b":
    print("The text is 'b'");

  case "B":
    continue isB;

  default:
    print("The text is unknown");
}
```



```
String text = "a";

print(switch(text) {
  "a" => "The text is 'a'";
  "b" => "The text is 'b'";
  _ => "The text is unknown";
});
```

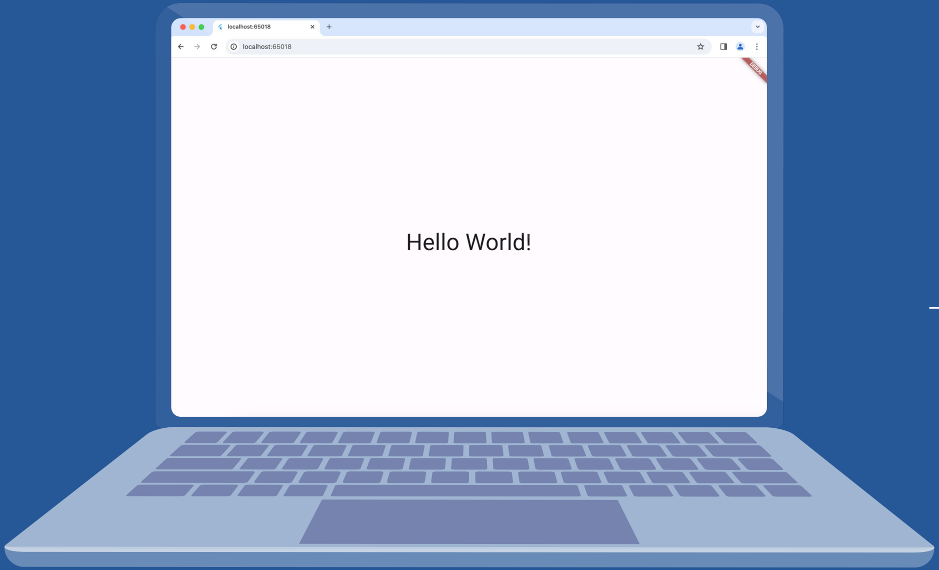


```
try {
  // ...
} on IOException catch(error, stacktrace) {
  // ...
} catch (error, stacktrace) {
  // ...
} finally {
  // ...
}
```



```dart
String result = asyncFunction()
  .then((res) => /* ... */)
  .catch((err) => /*  ...  */);

// OR

void doSomething async {
  try {
    String result = await asyncFunction();
    //  ...
  } catch(error, stacktrace) {
    //  ...
  }
}
```

https://dart.dev/language/async



Praxis

Installation fertigstellen

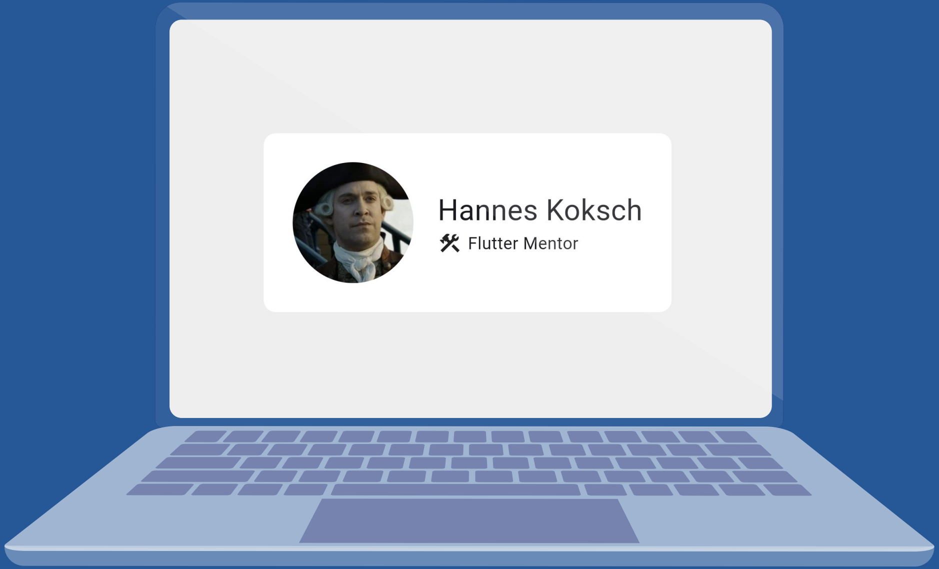
→ https://docs.flutter.dev/get-started/install

```
> flutter doctor
```

Hello World App erstellen

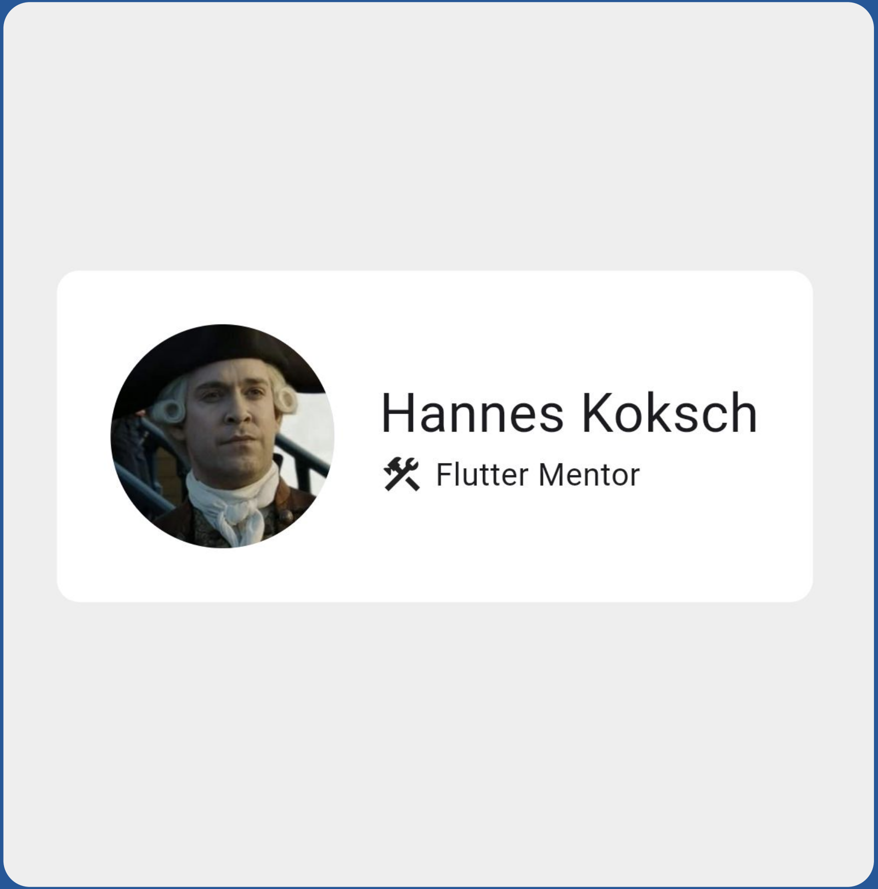
```
> flutter create --empty --platforms web
```

“Everything is a Widget”



Praxis

UI Nachbauen


Hannes Koksch

🛠 Flutter Mentor

Widgets für diese Aufgabe:

- Center
- CircleAvatar
- Column & Row
- Container
- Icon(Icons.xy)
- NetworkImage
- SizedBox / Padding
- Text
- Scaffold(backgroundColor: Colors.grey[200], ...

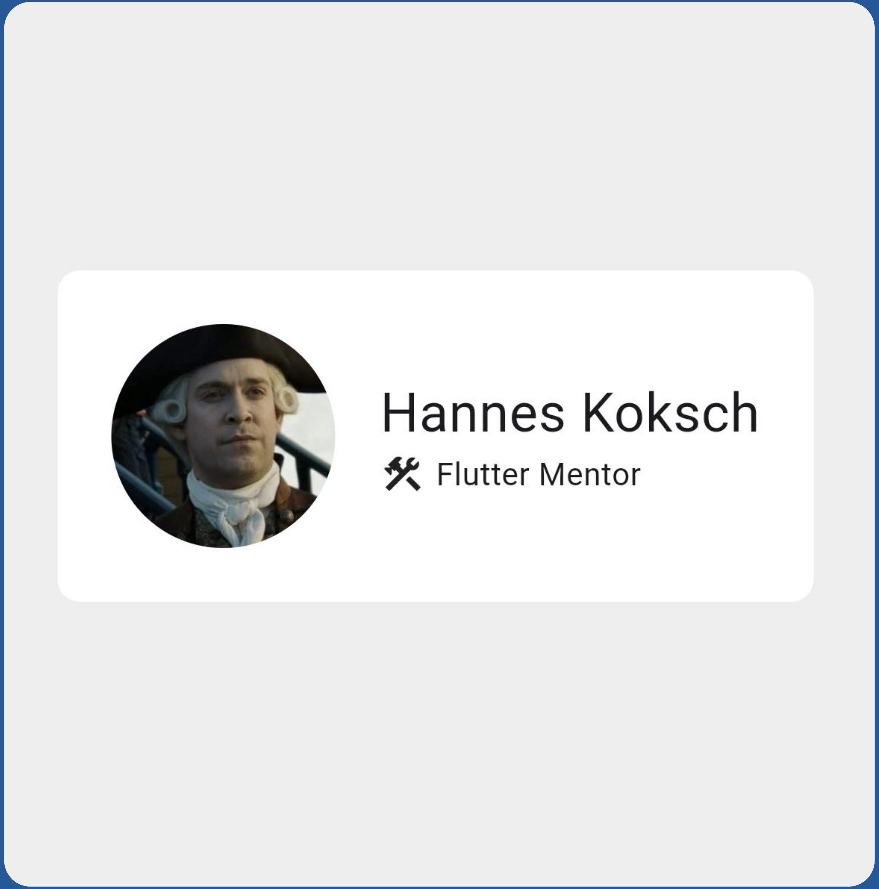
Ressourcen

Flutter Docs
https://docs.flutter.dev/

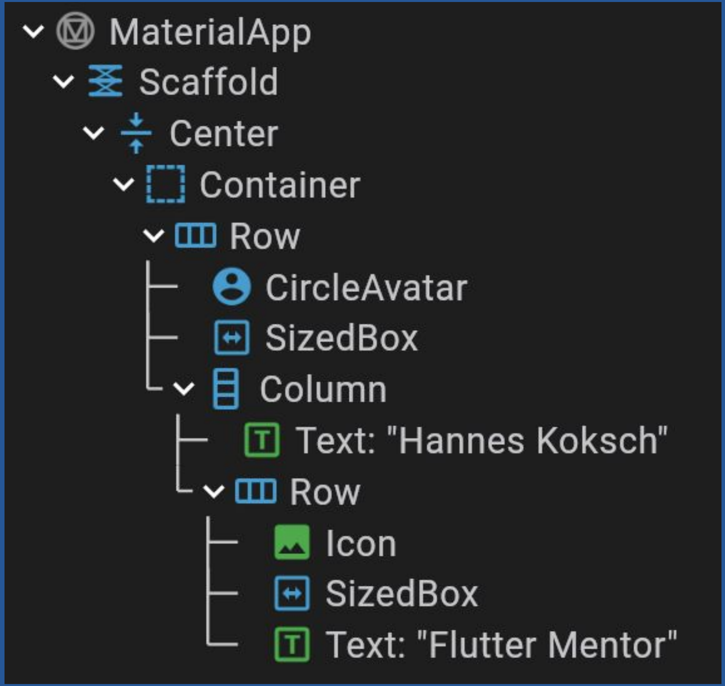
Widget Catalog
https://docs.flutter.dev/ui/widgets

Dart Docs
https://dart.dev/guides

Für Packages: Pub.dev
https://pub.dev/



Widget-Tree



```
MaterialApp
  Scaffold
    Center
      Container
        Row
        ├ CircleAvatar
        ├ SizedBox
        └ Column
          ├ Text: "Hannes Koksch"
          └ Row
            ├ Icon
            ├ SizedBox
            └ Text: "Flutter Mentor"
```

Lösungen zu Aufgaben und Beispiele aus der Vorlesung auf Gitlab:

https://gitlab.mi.hdm-stuttgart.de/mt098/fluttination

🏁 Ende des ersten Block 🏁