# 5.6.2 Asynchronität + Funktionen - Wiederholung

Aufgaben

Heute soll Asynchronität nochmal geübt werden. Dies soll zuerst in Dart geschehen und anschließend in Flutter.

## Vorgehen

1. Gruppiere die Funktionen logisch in Projekten (alle Funktionen, die etwas auf der Konsole ausgeben, alle Funktionen, die etwas mit zwei Zahlen berechnen, ...).
2. Erstelle die passenden Ordner und Dateien in dem Projekt (Projektarchitektur).
3. Schreibe dann den Code.

## Hinweise

- Du kannst die Funktionen erst einmal ohne **async** erzeugen, bevor du sie dann auf **async** umstellst.
- Denke an die **async**-Gebote:
    - Wenn ein **await** verwendet werden soll, muss die Funktion, die es enthält, **async** sein.
    - Eine **async**-Funktion muss ein **Future** zurückgeben[1].
    - Ohne **await** wird nicht gewartet, sondern die Funktion läuft *gleichzeitig*.
    - **await** packt ein Future, das von einer Funktion zurückgegeben wird, aus.
- Versuche den Code für die Aufgaben selber zu schreiben, anstatt externe Hilfe oder eine vorhandene Funktion zu verwenden. Dies hilft dir, die Vorgehensweise und das Mindset eines Programmierers zu üben.
- Wenn du eine Funktion bereits selber geschrieben hast, kannst du diese natürlich in darauf folgenden Funktionen benutzen. Manche Funktionen bauen sogar auf vorherigen Funktionen auf.
- Alle Funktionen, die nicht explizit etwas auf der Konsole **ausgeben** sollen, sollen etwas **zurückgeben**, also einen Rückgabewert haben, der nicht void ist.
- Bei Aufgaben mit Zahlen sind in der Regel ganze Zahlen gemeint, außer die Aufgabe fordert etwas anderes.
- Die Gestaltung der App ist euch überlassen, also wie die UI aussieht. Es ist aber empfehlenswert, zuerst die Funktionalität zuende zu machen, bevor man die UI aufhübscht 😀

[1] Kann auch void zurückgeben, ich weiß 🙂Immer **Future<void>** zu verwenden ist aber die sicherere Wahl und man kann es anders machen wenn man weiß warum.

# Aufgaben in reinem Dart

## Hello World

Schreibe eine Funktion, die nach einer Verzögerung "Hello, World!" *auf der Konsole* ausgibt.

## Hello $Name

Schreibe eine Funktion, die einen Namen entgegen nimmt und diesen nach einer Verzögerung mit "Hello " davor *auf der Konsole* ausgibt.

## Summe von zwei Zahlen

Schreibe eine Funktion, die die Summe von zwei Zahlen berechnet und das Ergebnis nach einer Verzögerung zurückgibt.

## Finde die größere Zahl

Schreibe eine Funktion, die die größere von zwei Zahlen nach einer Verzögerung zurückgibt.

## Gerade Zahl

Schreibe eine Funktion, die prüft, ob eine übergebene Zahl gerade ist und das Ergebnis nach einer Verzögerung zurückgibt.

# Aufgaben als Flutter-App

- Die meisten Werte, die den Funktionen in diesen Aufgaben übergeben werden, können entweder hardkodiert sein (fest im Code der App) oder es kann ein **TextField** für die Eingabe geben, falls du dich schon damit sicher fühlst.
- Die Berechnung soll verzögert sein, als käme sie von einer externen API. Die entsprechende Funktion kann entweder nach dem Starten der App ausgelöst werden oder durch einen Button.

## Summe

Schreibe eine App, die die Summe einer Liste von Zahlen anzeigt.

## Durchschnitt

Schreibe eine App, die den Durchschnitt einer Liste von Zahlen berechnet. Der Durchschnitt kann auch eine Kommazahl sein (Bsp.: [2, 3] -> 2.5).

## Häufigkeit Buchstabe

Schreibe eine App, die zurückgibt, **wie oft** ein Buchstabe in einem String vorkommt.

## Finde den Buchstaben

Schreibe eine App, die zurückgibt, **ob** ein Buchstabe in einem String vorkommt.

## Vorzeichen

Schreibe eine App, die prüft, ob eine Zahl negativ, positiv oder 0 ist. Überlege, wie die Rückgabe aussehen könnte und welche verschiedenen Möglichkeiten es gibt.

## Aufteilung

Schreibe eine App, die eine Zeichenkette in ihre einzelnen Zeichen aufteilt und diese zurückgibt.

## Buchstabenzahl

Schreibe eine App, die für eine Liste aus Zeichenketten zurückgibt, wie viele Zeichen jede der Zeichenketten hat. Der Rückgabewert soll jede Zeichenkette und die Anzahl der Zeichen darin enthalten. (Bsp: “David” -> 5, “Angi” -> 4 etc.)

**Viel Spaß bei der Bearbeitung der Aufgaben!** 🥳

# Bonusaufgaben

Diese Aufgaben können gemacht werden, wenn man schnell fertig ist oder eine größere Herausforderung sucht. Sie sind mit den Dart-Programmierkonstrukten lösbar, die wir bereits kennen gelernt haben.

## Textanalyse

Schreibe eine App, die eine Zeichenkette akzeptiert und die Anzahl der Leerzeichen, Vokale und sonstigen Zeichen zurückgibt.

## FizzBuzz

Schreibe eine App, die die Zahlen von 1 bis zu einer übergebenen Zahl *auf der Konsole ausgibt*. Bei Vielfachen von 3 soll "Fizz" und bei Vielfachen von 5 "Buzz" ausgegeben werden. Bei Vielfachen von 3 *und* 5 soll stattdessen "FizzBuzz" ausgegeben werden.

## Quadratmuster

Schreibe eine App, die ein Muster in Form eines Quadrates *auf der Konsole* ausgibt, wobei die Größe des Quadrates eine übergebene Zahl ist. Als Zeichen für das Muster kann zum Beispiel '#' verwendet werden.
*Hinweis*: Wenn man ein Leerzeichen vor die '#' macht, sieht es eventuell besser aus.
*Hinweis 2*: Für diese Aufgabe ist `stdout.write(...)` sehr hilfreich.

## Palindrom

Schreibe eine App, die überprüft, ob eine übergebene Zeichenkette ein Palindrom ist. Ein Palindrom ist ein Wort, das von hinten wie vorn vorne gelesen dasselbe Wort ist. Beispiele: Hannah, neben, Otto, Rentner.
*Hinweis*: Diese Aufgabe kann mit vorhandenen Mitteln gelöst werden oder mit verschiedenen, in Dart eingebauten Funktionen.

## Klammerproblem

Schreibe eine App, die prüft, ob in einem Ausdruck mit Klammern jede geöffnete Klammer eine passende schließende Klammer hat: "(a + b) * ((c + d) * e)".
Weitere Beispiele:

- `"(((())()))"` ist gültig, da jede Klammer passend geschlossen wird.
- `"(())))"` ist *nicht* gültig, da eine Klammer mehr geschlossen wird als geöffnet.
- `"((()()"` ist *nicht* gültig, da eine Klammer mehr geöffnet wird als geschlossen.
- `")()"` ist nicht gültig, da die erste Klammer keine öffnende Klammer hat.
- `"())()(()"` ist nicht gültig, da eine Klammer geschlossen wird, die keine öffnende Klammer hat und eine öffnende Klammer keine passende schließende Klammer hat.

*Hinweis*: Auch hier gibt es wieder verschiedene Lösungsansätze.