Exception Handling in Dart

Module 5 , Chapter 1

Introduction

Exceptions are unexpected events (like wrong input, missing files, or divide-by-zero) that interrupt program flow.

Exception Handling ensures your program doesn't crash — it helps handle errors gracefully.

Benefits:

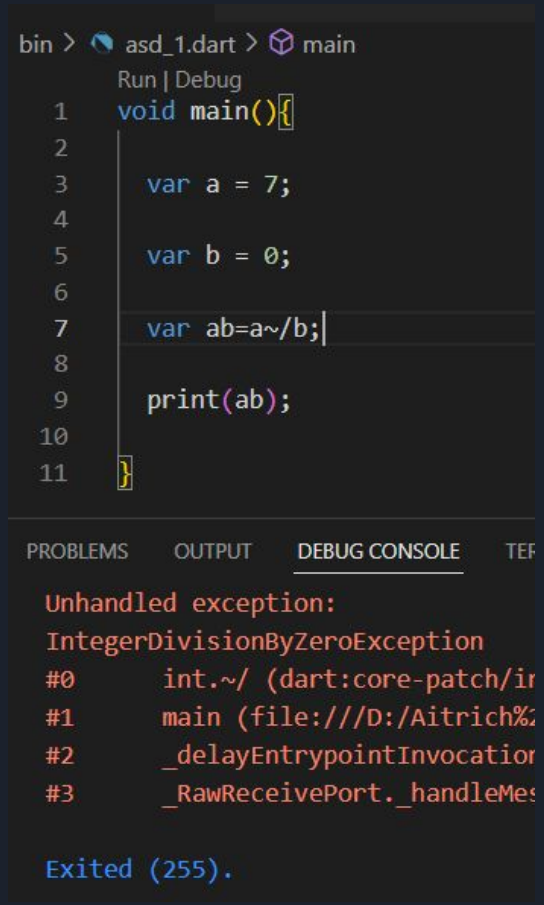
- Prevents crashes
- Easier debugging
- Better user experience
- More reliable and maintainable code

What is an Exception?

Example: dividing a = 7 by b = 0 → causes an error (“Division by zero”).

Such runtime problems are called exceptions.

Handling these is called exception handling.

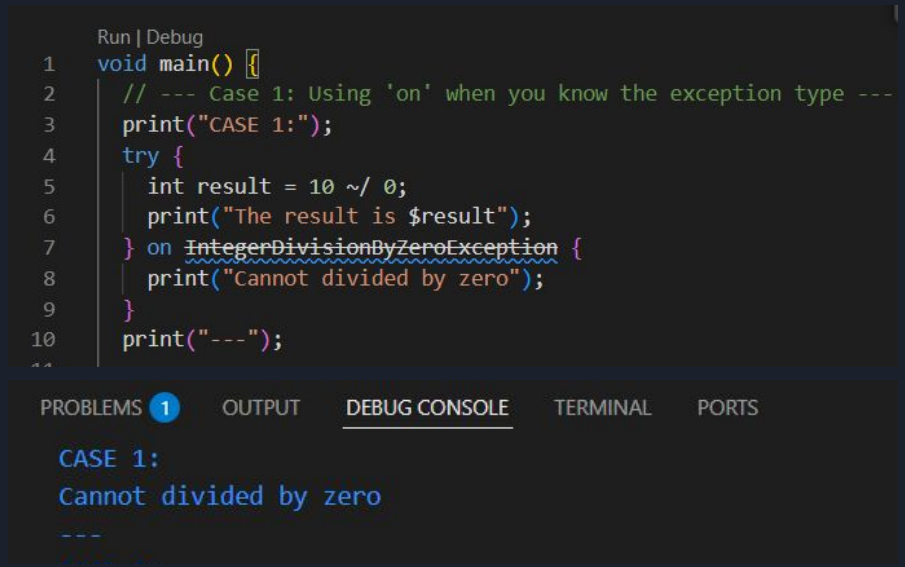


Handling Exceptions in Dart

(try , catch , on , finally)

try: contains code that might cause an exception.

on: handles a specific type of exception.



catch: handles unknown exceptions.

finally: runs always, whether an exception occurs or not.

Example cases:

Case 1: You know the exception → use on.

Case 2: You don't know the exception → use catch.

Case 3: Always want code to run → use finally.

```
12  // --- Case 2: Using 'catch' when you don't know
13  // the exception or for any exception ---
14  print("CASE 2:");
15  try {
16    int result = 10 ~/ 0;
17    print("The result is $result");
18  } catch (e) {
19    print("The expection throw is $e");
20  }
21  print("---");
22
23  // --- Case 3: Using 'finally' for cleanup code
24  // (runs whether an exception is thrown or not) ---
25  print("CASE 3:");
26  try {
27    int result = 10 ~/ 0;
28    print("The result is $result");
29  } catch (e) {
30    print("The expection throw is $e");
31  } finally {
32    print("This is finally clause");
33  }
34 }
```

```
CASE 2:
The expection throw is IntegerDivisionByZeroException
---
CASE 3:
The expection throw is IntegerDivisionByZeroException
This is finally clause

Exited.
```

No.	Exception	Description
1	FormatException	Data cannot be parsed due to incorrect format (e.g., text instead of number).
2	IntegerDivisionByZeroException	Occurs during integer division (~) by zero.
3	RangeError	Index or value is outside a valid, specific range (e.g., list index out of bounds).
4	TimeoutException	An asynchronous operation (Future) failed to complete before its deadline.
5	DeferredLoadException	A lazily loaded library failed to load at runtime.
6	IOException	Base class for exceptions related to file or network Input/Output operations.

Custom Exceptions

- Created when built-in exceptions don’t cover your specific case.
  Example: detecting device overheating.
- Uses these keywords:
  - class: defines a new exception type.
  - constructor: allocates memory for the object.
  - implements: defines how a class fulfills an interface.
  - throw: used to manually raise an exception.

```
Run | Debug
1  void main(){
2
3    try{
4      throEx();
5    }
6    on CustomEx catch(e){
7      print(e.a);
8    }
9  }
10
11 void throEx(){
12   throw CustomEx('Custom Exception');
13 }
14
15 class  CustomEx implements Exception{
16   String a;
17   CustomEx(this.a);
18 }
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

Custom Exception

Exited.
```

Exercise

```
PROBLEMS 4   OUTPUT   DEBUG CONSOLE   TE
format is incorrect

Exited.
```

```
String x='ai';
  int y=6;
```

```
PROBLEMS 4   OUTPUT   DEBUG CONSOLE   TERMINAL
IntegerDivisionByZeroException occurred!'

Exited.
```

```
String x='5';
  int y=0;
```

```
PROBLEMS 4   OUTPUT   DEBUG CONSOLE   TERMINAL
2
result lessthan 5

Exited.
```

```
String x='6';
  int y=3;
```

```
Run | Debug
1  void main(){
2    String x='123';
3    int y=5;
4    try{
5      int xx = int.parse(x);
6      var result=xx~/y;
7      print(result);
8      if(result<5){
9        trowExcep();
10     }}
11   on CustomExcep catch (e){
12     print(e.a);
13   }
14   on FormatException{
15     print("format is incorrect");
16   }on IntegerDivisionByZeroException{
17     print("IntegerDivisionByZeroException occurred!'");
18   }
19 }
20 void trowExcep(){
21   {
22   throw CustomExcep('result lessthan 5');
23   }
24 }
25 class CustomExcep implements Exception{
26   String a;
27   CustomExcep(this.a);
28 }

'ROBLEMS 4   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS
24

Exited.
```