Error Handling in Dart

Error handling in Dart involves the mechanisms used to deal with exceptions and errors that occur during program execution. To avoid stoppage of the program Dart provides several constructs for error handling:

First way-:

try-catch: This is the most common way to handle exceptions in Dart. You enclose the code that might throw an exception within a **try** block, and then catch and handle any exceptions that occur using one or more **catch** blocks. For example:

```
        try {
  // code that might throw an exception
  var result = 10 ~/ 0; // Division by zero will throw an exception
} catch (e) {
  // handle the exception
  print('An error occurred: $e');
}
```

Second way-:

on: You can use **on** to catch specific types of exceptions. This allows you to handle different types of exceptions differently. For example:

```
try {
  // code that might throw an exception
  var result = 10 ~/ 0; // Division by zero will throw an exception
} on IntegerDivisionByZeroException {
  // handle division by zero exception
  print('Cannot divide by zero');
} catch (e) {
  // handle other exceptions
  print('An error occurred: $e');
}
```

Third way-:

finally: The finally block is used to execute code regardless of whether an exception is thrown or not. This block is often used for cleanup tasks, such as closing files or releasing resources. For example:

```
try {
  // code that might throw an exception
} finally {
  // cleanup code
}
```

Forth way-:

throw: You can manually throw exceptions using the **throw** keyword. This allows you to raise exceptions programmatically when certain conditions are met. For example:

```
void depositMoney(int amount) {
  if (amount <= 0) {
    throw ArgumentError('Amount must be greater than zero');
  }
  // deposit money logic
}
```

Fifth way-:

Custom Exceptions: You can define your own exception classes by extending the **Exception** class or one of its subclasses. This allows you to create custom exceptions that provide more information about specific error conditions in your application. For example:

```
class MyCustomException implements Exception {
  final String message;

  MyCustomException(this.message);
}
```

```
void myFunction() {
  // code that might throw a custom exception
  throw MyCustomException('An error occurred in myFunction');
}
```