Error handling

Error handling in Dart is done using `try`, `catch`, and `finally` blocks. Here's a basic overview of how you can use them:

1. **Try Block**: Contains the code that might throw an error.
2. **Catch Block**: Handles the error if one occurs.
3. **Finally Block**: Contains code that runs regardless of whether an error occurred.

Here's an example:

```
void main() {
  try {
    // Code that might throw an error
    int result = 10 ~/ 0; // This will throw a DivisionByZeroError
    print(result);
  } catch (e) {
    // Code that runs if an error occurs
    print('An error occurred: $e');
  } finally {
    // Code that always runs
    print('This always runs.');
  }
}
```

Types of Errors

1. **Exception**: An object that represents an error or exceptional condition.
2. **Error**: Represents a problem that is generally not meant to be caught, such as `OutOfMemoryError`.

Catching Specific Exceptions

You can catch specific types of exceptions:

```
void main() {
  try {
    // Code that might throw an error
    int result = 10 ~/ 0; // This will throw a DivisionByZeroError
    print(result);
  } on IntegerDivisionByZeroException {
    // Handle specific exception
    print('Cannot divide by zero.');
  } catch (e) {
    // Handle all other exceptions
    print('An error occurred: $e');
  }
}
```

Custom Exceptions

You can define your own exceptions by extending the `Exception` class:

```
class CustomException implements Exception {
  final String message;
  CustomException(this.message);

  @override
  String toString() => 'CustomException: $message';
}

void main() {
  try {
    throw CustomException('Something went wrong.');
  } catch (e) {
    print(e);
  }
}
```

This is a basic introduction, but Dart’s error handling allows for quite a bit of flexibility depending on what you need!

A much simpler example:

```
void main() {
        try{
                // code you want to excute. and think it may contain errors
                int x = 20 ~/ 0;
                // infinity //try and catch block won't crash the programe and
continue                the code
```

```
                print("the result is $x");
        } catch (e, s) {
        // name whatever you what in this case I want the error named "e"
                print("the error is $e"); // for production -> print the error
                print("the stack is $s"); // for debugging -> print the stack
which means           the detailed reason for the error
        } finally {
                print("this code will always run");
        }

        print("");
        print("Continue the code");
}
```

```
//extending class Age
// with Exception class
class Exception{
  String error() => "ERROR";
}

class Age implements Exception{
  @override
        String error() => "error, you are less than 15";
}

void main(){
        int age_1 = 20;
        int age_2 = 14;

        try{
        // checking age and calling if the exception occur
        check(age_1);
        check(age_2);
        } catch(e) {
                print("you can't enter our plateform");
        }

}

//checking age
void check(int age){
        if(age < 15){
                throw new Age();
        } else {
                print("you can enter our plateform");
  }
}

//OUTPUT
// you enter our plateform
// you can't enter our plateform
```