Generics in Flutter and Streams in Flutter

1. Generics in Flutter

Generics allow you to define classes, methods, and functions with placeholder types.

This enables type safety while allowing the code to be more flexible and reusable.

Example of Generics:

Suppose you want to create a class that can hold any type of data, whether it's an integer,

string, or custom object. Here's how you can use generics:

```
class Box<T> {
  T content;
  Box(this.content);
  void displayContent() {
    print("Content: $content");
  }
}

void main() {
  var intBox = Box<int>(10);
  intBox.displayContent();  // Output: Content: 10

  var stringBox = Box<String>("Hello, Generics!");
  stringBox.displayContent();  // Output: Content: Hello, Generics!
}
```

In this example, T is a generic type parameter that can be replaced with any data type.

The Box class can now hold content of any type, making it reusable and type-safe.

2. Streams in Flutter

Streams in Flutter are used for handling a sequence of asynchronous events. They provide a way to receive a series of data over time, such as data from a web request or user inputs, without blocking the main thread.

Example of Streams:

```
import 'dart:async';
void main() {
  final StreamController<int> controller = StreamController<int>();
  controller.add(1);
  controller.add(2);
  controller.add(3);

  controller.stream.listen((data) {
    print("Received: $data");
  });
  controller.close();
}
```

In this example, StreamController is used to create a stream. We add data to the stream using controller.add(). We listen to the stream using controller.stream.listen().

Combining Generics and Streams:

```
import 'dart:async';
Stream<T> createStream<T>(List<T> items) async* {
  for (T item in items) {
    yield item;
  }
}

void main() {
  Stream<int> intStream = createStream<int>([1, 2, 3, 4, 5]);
  intStream.listen((data) {
    print("Received int: $data");
  });

  Stream<String> stringStream = createStream<String>(["A", "B", "C"]);
  stringStream.listen((data) {
    print("Received string: $data");
  });
}
```

In this example, createStream<T>() returns a Stream of any type, handling both int and String streams.