In Dart, null safety is a feature that helps us write more robust and predictable code by making it explicit whether a variable can be null or not. Null safety was introduced in Dart 2.12, and it includes several operators and features to work with nullable types. Here are some key null safety operators in Dart:

## 1. **Null-aware operators (`?.` and `??`):**

- ?. (null-aware access): Allows you to call a method or access a property only if the object is non-null. If the object is null, the expression evaluates to null.

```
var nameLength = user?.name?.length;
```

- ?? (null-aware coalescing operator): Returns the expression on its left if it's non-null; otherwise, it returns the expression on its right.

```
var userName = user?.name ?? "Guest";
```

## 2. Null assertion operator (!):

- The null assertion operator asserts that its operand is non-null and allows you to treat a nullable expression as non-nullable.

```
var length = user!.name.length;
```

## 3. Null safety with collections (**List**, **Set**, **Map**):

- Dart null safety introduces the `?` suffix for declaring nullable collections.

```
List<int>? numbers;
```

- The `..` cascade operator now works with nullables:

```
user?.settings..updateTheme("Dark")..updateFontSize(16);
```

## 4. Late Keyword:

- The late keyword is used to declare variables that will be initialized later, allowing you to use non-nullable types with deferred initialization.

```
late String name;
```

## 5. Null safety with functions:

- Function parameters can be explicitly marked as nullable using `Type? paramName`.

```
void printName(String? name) {
  print(name ?? "Name is null");
}
```

- Return types can also be nullable using `Type?`.

```
String? findName() {
  // ...
}
```