

Null safety:

Null safety is a feature introduced in Dart 2.6 that helps prevent null-related errors at compile time. It enforces the types of variables and expressions, ensuring that you can't accidentally access a variable that might be null. This leads to more robust and predictable code.

1. Non-nullable types:

- By default, variables are non-nullable, meaning they cannot hold null values. This is enforced by specifying the type explicitly:

Example:

```
String name = "John"; // Name cannot be null
int age = 30; // Age cannot be null
```

2. Nullable types:

- You can explicitly mark a type as nullable using the `?` operator. This allows the variable to hold either a value of the specified type or null:

Example:

```
String? nickname; // Nickname can be null or a String
nickname = "Jack"; // Ok, setting a String value
nickname = null; // Ok, setting null
```

Null-aware operators

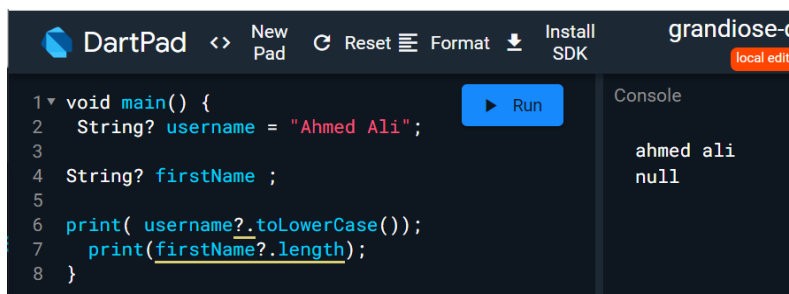
Null-aware operators are your best friends in Dart when dealing with nullable types and preventing null-related errors. They offer concise and safe ways to access properties, call methods, and perform operations on nullable values. Here's a breakdown of the most common ones:

Operator	Meaning
<code>??</code>	If-null operator
<code>??=</code>	Null-aware assignment operator
<code>?.</code>	Null-aware access & method invocation operator
<code>!</code>	Null assertion operator
<code>?..</code>	Null-aware cascade operator

<code>?[]</code>	Null-aware index operator
<code>...?</code>	Null-aware spread operator

1. ?. (null-safe access operator):

This operator safely accesses properties of a potentially null object. If the object is null, it returns null instead of throwing an error.



```

1 void main() {
2   String? username = "Ahmed Ali";
3
4   String? firstName;
5
6   print( username?.toLowerCase());
7   print(firstName?.length);
8 }

```

Console output:

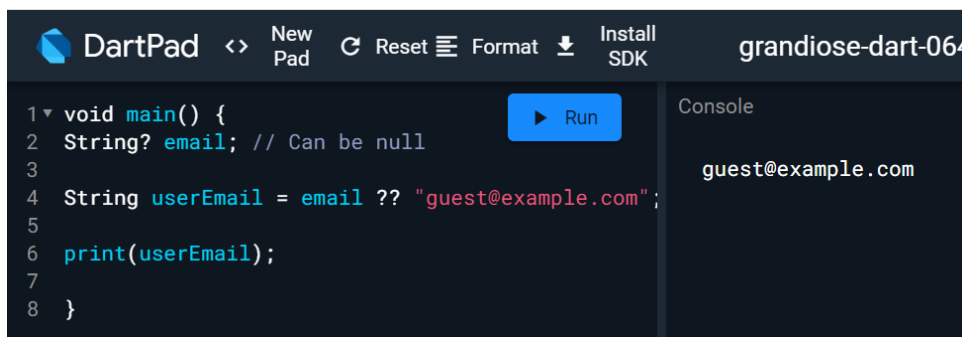
```

ahmed ali
null

```

2. Null-coalescing operator (??):

This operator provides a default value if the left operand is null. It's useful for setting default values in case of null values.



```

1 void main() {
2   String? email; // Can be null
3
4   String userEmail = email ?? "guest@example.com";
5
6   print(userEmail);
7
8 }

```

Console output:

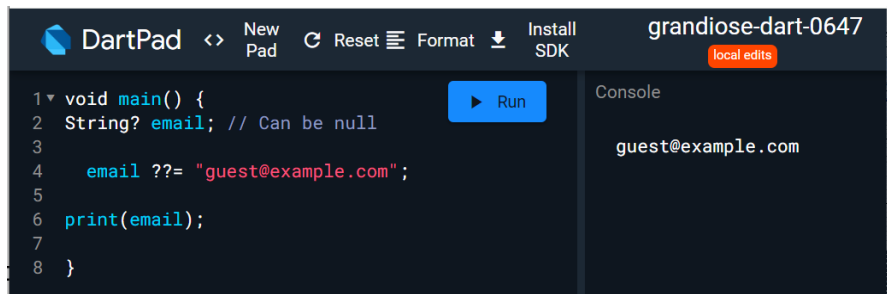
```

guest@example.com

```

3. Null-aware assignment operator (??=):

This operator assigns a value to a variable only if it's null. It's handy for initializing variables based on conditions.



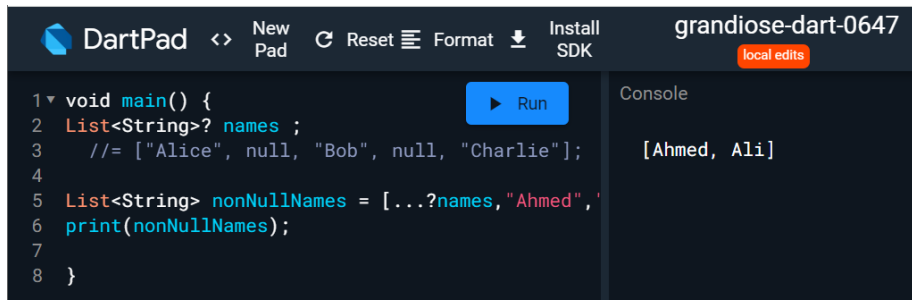
The screenshot shows the DartPad interface with a file named 'grandiose-dart-0647'. The code in the editor is as follows:

```
1 void main() {  
2   String? email; // Can be null  
3  
4   email ??= "guest@example.com";  
5  
6   print(email);  
7  
8 }
```

The 'Run' button is highlighted. The console on the right displays the output: 'guest@example.com'.

4. Null-aware spread operator (...?)

This operator safely spreads elements from an iterable that might contain null values. It skips null elements and only spreads non-null ones.

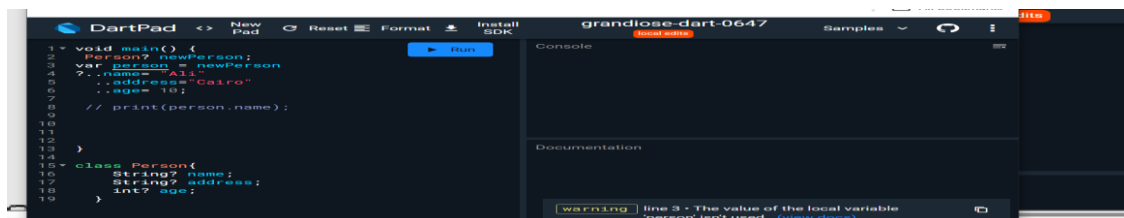


The screenshot shows the DartPad interface with a file named 'grandiose-dart-0647'. The code in the editor is as follows:

```
1 void main() {  
2   List<String>? names ;  
3   // = ["Alice", null, "Bob", null, "Charlie"];  
4  
5   List<String> nonNullNames = [...?names, "Ahmed", '  
6   print(nonNullNames);  
7  
8 }
```

The 'Run' button is highlighted. The console on the right displays the output: '[Ahmed, Ali]'.

5. Null-aware cascade operator (?..)



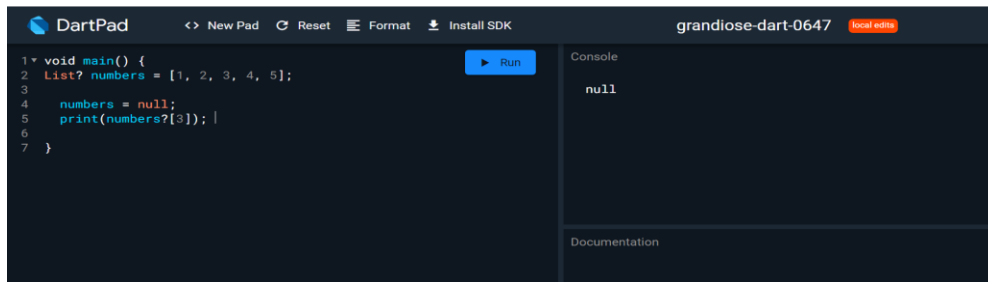
The screenshot shows the DartPad interface with a file named 'grandiose-dart-0647'. The code in the editor is as follows:

```
1 void main() {  
2   Person? newPerson;  
3   var person = newPerson  
4   ?..name = "Ali"  
5   ..address = "Cairo"  
6   ..age = 10;  
7  
8   // print(person.name);  
9  
10  
11  
12  
13  
14  
15 class Person {  
16   String? name;  
17   String? address;  
18   int? age;  
19 }
```

The 'Run' button is highlighted. The console on the right is empty. A warning message is displayed at the bottom: 'Warning line 8 - The value of the local variable 'person' isn't used. (View docs)'.

6-Null-aware index operator(?[])

The null-aware index operator `?[]` allows you to access an element of a list when the list might be null



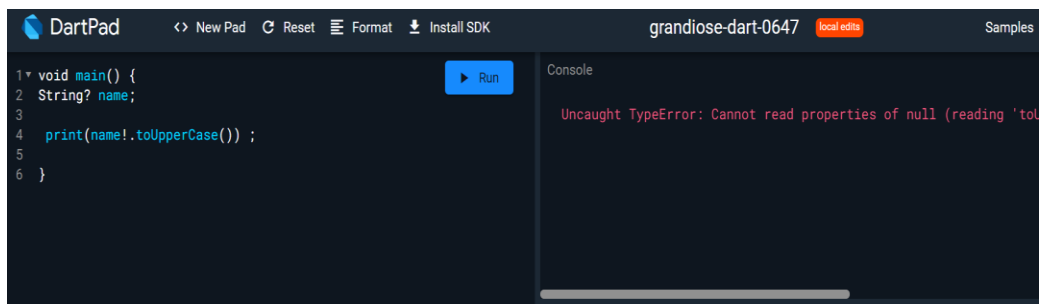
The screenshot shows the DartPad interface with a code editor on the left and a console on the right. The code in the editor is as follows:

```
1 void main() {  
2   List? numbers = [1, 2, 3, 4, 5];  
3  
4   numbers = null;  
5   print(numbers?[3]);  
6  
7 }
```

The console on the right displays the output: `null`.

7- Null assertion operator (!)

The `!` operator tells the compiler that you are sure a variable or expression is not null and allows you to access its properties and methods directly, bypassing null checks.



The screenshot shows the DartPad interface with a code editor on the left and a console on the right. The code in the editor is as follows:

```
1 void main() {  
2   String? name;  
3  
4   print(name!.toUpperCase());  
5  
6 }
```

The console on the right displays an error message: `Uncaught TypeError: Cannot read properties of null (reading 'toUpperCase')`.