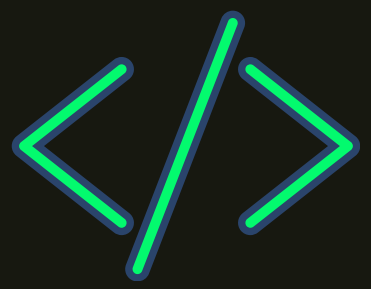


Flutter & Dart

systems

What is Dart ?

Questions Questions

Dart is a client-optimized language for developing fast apps on any platform.

Its goal is to offer the most productive programming language for multi-platform development.

Origin of Dart

Release: Nov 13

- Dart is a programming language developed by **Google**.
- It was first unveiled at the GOTO conference in **Aarhus, Denmark, in October 2011.**
- Dart 1.0 was released in **November 2013.**
- Dart 2.0, a **major update**, was released in **August 2018.**

Flutter is the UI Framework of Dart.

Features of Dart

- Object Oriented
- Strongly Typed
- Auto Garbage Collection
- Mixins
- Single Inheritance
- Asynchronous Programming
- Isolates

@theakhilsarkar

What is Flutter ?

main.dart

Flutter is an open-source UI software development toolkit developed by Google.

Flutter was released in May 2017.

#Flutter #Dart #Programming #NeverGiveUp

Flutter

The project was initially known as the "**Sky**" project and was officially announced as Flutter in **2015**.

Key Years

- The first stable version, Flutter 1.0, was released in December 2018.
- The current stable version is Flutter 3.19 on 15 Feb 2024.

Features

- Hot Reload
- Widget's Library
- Cross Platform Development
- Dart
- Expressive UI and Customization

Advantages

- Single Codebase
- High Performance
- Fast Development
- Rich Widget Library
- Documentation Support
- Growing Community

@theakhilsarkar

Dart's compiler technology

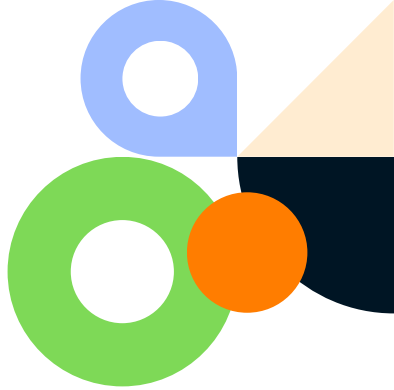
Native Platform

- For apps targeting mobile and desktop devices, Dart includes both a Dart VM with just-in-time (JIT) compilation and an ahead-of-time (AOT) compiler for producing machine code.

Get organized

- For apps targeting the web, Dart can compile for development or production purposes. Its web compiler translates Dart into JavaScript.

Operators in Dart



main.dart

- In Dart, operators are special symbols or keywords that perform operations on one or more operands.

- Perform mathematical or logical operations, and facilitate various computations within a program.

#Flutter #Dart #Programming #NeverGiveUp

Arithmetic Operators

main.dart

- **+** : Add
- **-** : Substract
- ***** : Multiply
- **/** : Divide
- **%** : Get reminder
- **~/** : Divide and return integer
- **-expr** : reverse sign

```
print(10%3); // output: 1
print(7~/2); // output: 3
print(-(3-5)); // output: 2
```

Equality and relational operators

To compare two values,variables or objects.

main.dart

- == : Equal
- != : Not Equal
- >= : Greater than equal to
- <= : Less than equal to

main.dart

- > : Greater than
- < : Less than

Equality and relational operators

- `print(2 == 2); // output: true`
- `print(2 != 2); // output: false`
- `print(13 > 2); // output: true`
- `print(12 < 3); // output: false`
- `print(3 >= 3); // output: true`
- `print(2 <= 3); // output: true`

Type test operators

The as, is, and is! operators are handy for checking types at runtime.

main.dart

- **as** : Typecast
- **is** : True if the object has the specified type
- **is!** : True if the object doesn't have the specified type

```
int a = 10;
  • print(a is int);  // output: true
  • print(a is! int); // output: false
dynamic x = 18;
int y = x as int;
  • print(y); // output: 18
```

Assignment operators

main.dart

- = : assign value
- += : assign value after add
- -= : assign value after subtract
- *= : assign value after multiply

main.dart

- %= : assign value after reminder
- /= : assign value after division
- ~/= : assign value after int division
- ??= : assign value if variable is null

Assignment operators

- `int a = 10; // a = 10`
- `a += 10; or a = a + 10; // a = 20`
- `a -= 10; or a = a - 10; // a = 10`
- `a *= 10; or a = a * 10; // a = 100`
- `a /= 10; or a = a / 10; // a = 5`
- `a ~/= 2; or a = a ~/ 2; // a = 2`
- `int? x; // output: true`
- `x ??= 10; // Assign the value only if it is null.`

Logical operators

To connect two or more conditions.

main.dart

- ! : Invert expression
- && : If both conditions are true then the answer is true.
- || : If any single condition is true then the answer is true.

- ! : Logical NOT
- && : Logical AND
- || : Logical OR

```dart
print(!(true)); // false
print((true)&&(true)); // true
print((false)||(true)); // true
```

Null Saftey operators

main.dart

- To specify that a variable type can have a null value, add a ? after the type annotation: int? i.

- ? : null-aware operator
- ! : null-check operator
- ?? : null-replacement operator

Null Saftey operators

main.dart

- `int? a; //a might be null`
- `print(a!); //use ! when you are sure a is not null.`
- `a ?? 10; // if ‘a‘ is null then 10 will return otherwise value of ‘a‘ will return.`

Datatypes in Dart

main.dart

- In Dart, Datatype is the type of data that you use in development.
- Numbers(**int, double, num**)
- **String**
- Boolean(**bool**)
- Collection(**Lists,Maps,Sets**)

#Flutter #Dart #Programming #NeverGiveUp

Numbers

- **int** : whole numbers (negative, positive, zero).
- **double** : point numbers up to 15 digits.
- **num** : int and double both.

@theakhilsarkar

String

```
• String name = ‘Name’;       // Name
• String fname = “name”;      // name
• String msg = “’world’”;     // ‘world’
• String cat = ‘Con’+’Cat’; // Concat
```

@theakhilsarkar

Boolean

- mostly used in conditional programming.
- can hold only true/false.
- bool isLogin = true;
- bool isCheck = false;

@theakhilsarkar

Lists in Dart

main.dart

- In Dart, an **ordered group of objects** is represented by a **comma-separated** list of values or expressions **enclosed in square brackets.**

- List l1 = [1,”hello”,12.34, true];
- List l1;
- l1 = [1,2,...];

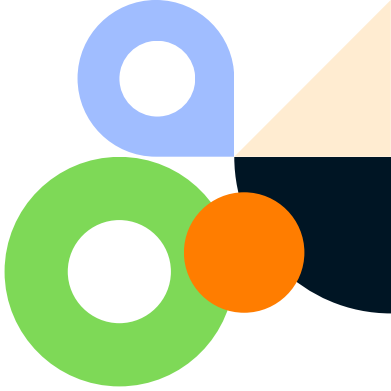
Insert in List

main.dart

```
• List list = [];
• list.add(value); // to add new element.
• list.addAll(List); // to add new list
• list.insert(index,value); // to add new
  element on index.
• list.insertAll(index,List); // to add
  new list on index
```

#Flutter #Dart #Programming #NeverGiveUp

Update in List

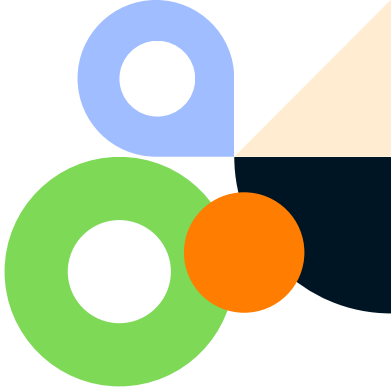


main.dart

```
• List list = [1,2,3,'dart',true];
• list[0] = 10; // [10,2,3,'dart',true]
• list[3] = 'Sky'; //
  [10,2,3,'Sky',true]
• list.replaceRange(start,end,List);
• list.replaceRange(0,3,[10,20,30]);
• // [10,20,30,3,'dart',true]
```

#Flutter #Dart #Programming #NeverGiveUp

Delete in List



main.dart

```
• List list = [1,2,3,'dart',true];
• list.remove(value); // to remove value
• list.removeAt(index); // to remove
  value at index.
• list.removeLast(); // to remove last
  value.
• list.removeRange(start,end);
```

#Flutter #Dart #Programming #NeverGiveUp

List Functions

```
list.reversed.toList(); // return
reversed List.

toList() is convert output into List.

list.getRange(start,end); //return List
from given range.
```

@theakhilsarkar

List Functions

```
list.firstWhere((i) => (condition));
//This method returns the first element
from the list when the given condition is
satisfied.

list.lastWhere((i) => (condition));
// return last element when given con..
```

@theakhilsarkar

List Functions

```
list.any((e) => e.contains('cricket'));

// This method returns a boolean depending
upon whether any element satisfies the
condition or not.
```

@theakhilsarkar

List Functions

```
list.take(count);

// This method returns iterable starting
from index 0 till the count provided from
given list.
```

@theakhilsarkar

List Functions

```
list.skip(count);

// This method ignores the elements
starting from index 0 till count and
returns remaining iterable from given
list.
```

@theakhilsarkar