OOPS Logical

(P) verify by pc

(1) Ex

```
int a = 10
int b = a++ + ++a
      = 10 + 12
```

b = 22

(2)

```
int a = 10;
int b = a++;

S.o.pln(b);
S.o.pln(a);
```

a [10] int b [10] int

b = 10
a = 11

(3)

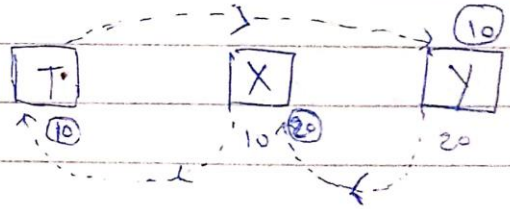
```
int a = 5
int b = ++a;

S.o.pln(b);
S.o.pln(a);
```

a [5] int b [6] int

b = 6
a = 6

(4) Swap 2 num using Temp var

X = 10, Y = 20




```
Temp = X    // 10
X = Y       // 20
Y = Temp    // 10
```

(5) Without third var Swap

a [10] b [20]

```
a = a+b    // 30
b = a-b    // 10
a = a-b    // 20
```

(6)

classmate
Date
Page

Programs

1. Fibonacci series

```
int n, i, n1 = 0, n2 = 1;
int nextnum = n1 + n2;

Printf (" Enter Number : ");
Scanf ("%d", &n);

Printf (" Fibonacci Series : %d, %d, ", n1, n2);

for (i = 3; i <= n; i++)
{
    Printf ("%d", nextnum);
    n1 = n2;
    n2 = nextnum;
    nextnum = n1 + n2;
}
```

O/P

```
Enter Number : 5
Fibonacci Series : 0, 1, 1, 2, 3
```

2. Largest number (Greatest number) among 5 numbers.

```
int a[10], i, max;
printf (" Enter 5 numbers : ");
    for (i = 0; i <= 4; i++)
        Scanf ("%d", &a[i]);
            max = a[0];
    for (i = 1; i <= 4; i++)
        if (max < a[i])
            max = a[i];
Printf (" greatest number is : %d ", max);
```

O/P

```
Enter 5 number : 34
53
66
23
8

Greatest number is : 66
```

Note:-

Find Quotient - $n_1 \sim/ n_2$ - integer division operator
Remainder - $n_1 \% n_2$ - modulo "

classmate
Date
Page

③ Palindrome number Ex 121

```
int originalnum = 12321;
int rev = 0;
int num = originalnum;

    while(num != 0) {
        int r = num % 10;
        rev = rev * 10 + r;
        num /= 10;
    }

if( originalnum == rev)
 {
    printf(" %d is a palindrome ", originalnum);
 }
else
 {
    printf(" %d is Not palindrome ", originalnum);
 }
```

O/P 12321 is a palindrome

④ Check number is Prime or not

```
int n, i; flag = 0;

printf(" Enter Number: ");
scanf(" %d ", &n);
```

```
for (i=2 ; i<=n/2 ; ++i)
    if(n % i == 0) {
    flag = 1 ;
    break ;
    }
if (n == 1) {
    Printf(" 1 is neighter Prime nor Composite ");
}
else
{
    if(flag == 0)
        Printf ("%d is a Prime number ", n);
    else
        Printf(" %d is not a Prime number ", n);
}
```

avoid if not chk 1

O/P

```
Enter Number : 7
7 is a Prime number.
```

(5) Reverse String

```
#include <string.h>
char str[100] = " Yash ";
Printf(" original string : %s \n", str);

int len = strlen(str);

for(int i = 0, j = len -1 ; i<=j ; i++, j--) {
    char c = str[i];
    str[i] = str[j];
    str[j] = c ; }
Printf(" Reverse String : %s", str);
```

original story : Yash

Reverse story : hsay

(6) Factorial Number

```
int fact(int n);
int main()
{
    int n;
    printf("Enter Number : \n");
    scanf("%d", &n);

    printf("\n Factorial of %d is %d \n", n, fact(n));

    return 0;
}
int fact(int n) {
    if (n == 0 || n == 1)
        return 1;
    Else
        return n * fact(n-1);
}
```

o/p

```
Enter Number :
5
factorial of 5 is 120
```

⑦ Find Largest and Second Largest number.

```
int a[10], i, max, second_max;

Printf (" Enter 5 Numbers : ");
    for(i = 0; i<5; i++)
        scanf("%d", &a[i]);
        max = a[0];
        second_max = a[1];

    for (i=1; i<5; i++) {
        if (max < a[i]) {
            second_max = max;
            max = a[i];
        }
        else if (second_max < a[i] && a[i] != max){
            second_max = a[i];
        }
    }
printf (" Greatest number is %d \n", max);
Printf (" Second 'greatest number is %d \n", second_max);

return 0; }
```

O/P

Enter 5 Numbers : 60
32
64
80
64
Greatest number is : 80
Second greatest number is : 64

(8) Armstrong Numbers

- is num that is equal to the Sum of cubes of it's digits

Ex 0, 1, 153, 370, 371 Armstrong Numbers.

Ex. nuee

0 1 List

```
final fixList = List<int>.filled(5, 0); // creates fixed
print(fixList); // [0,0,0,0,0]          length list
```

★ Constructors

1 empty

Create a new empty list

if growable is true, It create Empty list [].

if growable is false default, list of length zero.

E.g.

① final growableList = List.empty(growable: true); // []
growableList.add(1); // [1]

② final growableList = List.empty(growable: false); // []
growableList.add(1); // script error
Bcoz, we define growable: false that means can't be change

2 filled

- Create a list of the given length with fill at Each Position
- The length must be a non-negative Integer

E.g.

```
final zeroList = List<int>.filled(3, 0, growable: true); // [0,0,0]
```

→

for ex object
(abc)

Ex
```
final unique = List.generate(3, (_) => []); // [[],[],[]]
unique[0].add(499); // [[499], [], []]
```

- List.generate create list with fixed length and a New Object at each position.

4 Generates a list of values.

Ex
```
final growableList = List<int>.generate
                    (3, (int index) => index * index);
// [0, 1, 4]
```

→ Length must be non negative.

3 from

→ create a list containing all Elements.

5 of

→ creates a list from elements

6 unmodifiable

→ List can't have its Length or Elements changed.

Ex
```
final numbers = <int>[1,2,3];
final unmodifiableList = List.unmodifiable(numbers);
// [1,2,3]
unmodifiableList[1] = 87; // throws script Error.
```

Properties

1 first

→ A List.first is equivalent to List[0], both for getting and setting the value.

2 isEmpty

ex

```
final EmptyList = <int> [];
print ( EmptyList.isEmpty ) ; // true
```

→ ex `print ( EmptyList.iterator.moveNext() ) ; // false`

→ It is check is element in list next or not.

3 ~~isNotE~~ isNotEmpty

→ ex

```
final number = <int> {1, 2, 3};
print (number.isNotEmpty ) ; // true
print ( number.iterator.moveNext() ); // true.
```

4 Last

ex of final and const

5 Length

ex

```
final number = <int> [1, 2, 3] ;
Print (number.last ) ; //3
number . last = 10 ;
print ( number.last ) ; // 10
```

if i give const then it is not print it's give error

5 Length

E.x

```
final maybenumbers = <int?>[1, null, 3];
maybenumbers.length = 5
Print(maybenumbers); // [1, null, 3, null, null]
maybenumbers.length = 2;
print(maybenumbers); // [1, null]
```

E.x

```
final numbers = <int>[1,2,3];
numbers.length = 1;
Print(numbers); // [1]
numbers.length = 5; // throws, can't add Null's.
```

6 reversed

E.x

```
final numbers = <int>[1,2,3];
final revnum = numbers.reversed;
print(revnum.toList()); // [3,2,1]
```

METHODS

E.x

```
final n = <int>[1,2,3];
n.add(4); // [1,2,3,4]
n.addAll([4,5,6]); // 1,2,3,4,5,6
```

1 add.

2 addAll

3 any — check true, false and return — at least one element in the iterable satisfies condition

checks any element of this iterable satisfies test.

E.x

```
final n = <int>[1,2,3];
var re = n.any((element) => element >= 3); // true
```

4 asMap

E.x

```
var words = <String>['a', 'b', 'c'];
print(words); // [a,b,c]
var map = words.asMap();
Print(map); // {0:a, 1:b, 2:c}
Print(map.keys.toList()); // [0,1,2]
```

5 Cast

6 Clear
n.clear();
print(n.length); print(n); // 0 : []

7 Contains — check and return true, false.

E.x
final cib = <int, String> { 1: 'yash', 2: 'jay' };
final containone = cib.key.contains(1); // true,
final containyash = cib.values.contains('yash'); // true

8 elementAt — Returns the index th element

E.x
final numbers = <int> [1,2,3,4];
final elementat = numbers.elementAt(3); // 4

9 every — check every elent iterable test and return true & flale

E.x
final a = < double, String> { 0.06 : 'ab', 0.81 : 'cd' }
final every = a.key.every ((Key) => key < 1.0); // true
- if every element in the iterable satisfies the provide Condition, otherwise return false

10 Expand

E.x
(1)
var pairs = [[1,2], [3,4]];
var op = pairs.expand((pair) => pair).toList();
print(op); // [1,2,3,4]

and nested List ex:

(2)
List<List<int>> nestedlist = [[1,2], [3,4], [5,6]];
List <int> ab = nestedlist.expand((list) => list).toList();
print(ab); // [1,2,3,4,5,6]

11 fillRange — overwrites a range of elements with fillValue

E.x
final words = List.filled(5, 'obj');
print(words); // [obj, obj, obj, obj, obj]
words.fillRange(1, 3, 'new');
print(word); // [obj, new, new, obj, obj]
1 2 3 4 5

Here, (1, 3, 'new') That mean Start with one that mean after one(1) print "new" and end with 3 that means end cont in 3

addAll - add all elements of one collection to another collection.
It modifies the original collection

followedBy - It does not modify the original collection.

12 firstWhere

13 followedBy // Same as addAll.

Ex
```
var a = <String>['a', 'b', 'c'];
var updated = a.followedBy(['d', 'e']);
print(updated); // (a, b, c, d, e)
```

14 forEach - print all element, with new lines

Ex
```
final num = <int>[1, 2, 3];
num.forEach(print);
// 1
// 2
// 3
```

15 getRange - 0 ≤ Start ≤ end ≤ length
(Start: index, end: length)

Ex
```
final colors = <String>['red', 'blue', 'Green'];
final firstRange = colors.getRange(0, 2);
Print(firstRange.join(', ')); // red, blue
print(firstRange.join(':')); // red : blue

final Second = Color.getRange(1, 3);
Print(Second.join(', ')); // blue, Green
```

16 indexOf - Searches List of index
- if it's match the value it return Output 1. else -1

Ex
```
final notes = <String>['do', 're', 'mi', 're'];
print(notes.indexOf('re')); // 1
```
Return -1 if Element is not found
```
Print(notes.indexOf('ye')); // -1
```

17 indexWhere Search List of index

Ex
```
final notes = <String>['do', 're', 'mi', 're'];
final first = notes.indexWhere((note) => note.startsWith('r')); // 1
" Second = " " " " " " ('r'), 2); // 3
" index = " " " " " " ('k')); // -1
```

Ex indexOf, contains, lastIndexOf, indexWhere, lastIndexWhere

```
var myList = [0,2,4,6,8,2,8];
myList.contains(2); // true
  ".contains(5); // false
myList.indexOf(2); // 1
  ".lastIndexOf(2); // 5
myList.indexWhere((item) => item > 5); // 3
  ".lastIndexWhere((item) => item > 5); // 6
```

18 insert - insert Element at Position index in the list.

Ex
```
final numbers = <int> [1,2,3,4];
Const index = 2;
numbers.insert( index , 10);
Print (numbers); // [1,2,10,3,4]
```

19 insertAll

Ex above Example
```
final insertitems = [10,11];
numbers.insertAll(2, insertitems);
Print (numbers); // [1,2,10,11,3,4]
```

20 join - converts each Elements to a String and concatenates the String

Ex
```
final ab = <int, String> {1:'a', 2'b', 3'c'};
final joinedNames = ab.values.join('-'); // a-b-c
```

21 lastIndexOf - Search and match with list & return I or -I as o/p

Ex
```
final index = notes.lastIndexOf('re'); if in list // 1
  " index2 = " " " ('fa'); not in list // -1
```

22 lastIndexWhere

23 remove - return bool value - True or false

24 removeAt - remove perticulars Index value. Ex removeAt(2); Pass value in index value

25 removeLast - remove last obj. Ex removeLast();

26 removeRange - remove greater than or equal to Start and Less than End

Ex
```
final num = <int> [1,2,3,4,5];
num.removeRange(1,4);
print(num); // [1,5]
```

E.X where / firstWhere / lastWhere

```
var myList = [0,2,4,6,8,2,7];
myList.where((item) => item > 5).toList(); // [6,8,7]
myList.firstWhere((item) => item > 5); // 6
myList.lastWhere((item) => item > 5); // 7
```

27 removeWhere — bool test

E.X
```
final num = <String> ['one', 'two', 'three', 'four'];
num.removeWhere((item) => item.length == 3);
Print(num); // [three, four]
```

28 replaceRange — Replace Range of Elements, pass start with and End with range

E.X
```
final num = <int> [1,2,3,4,5];
final replacement = [6,7];
num.replaceRange(1,4, replacement);
print(num); // [1,6,7,5]
```

29 retainWhere — Remove all obj from this list that fail to satisfy test

E.X
```
final num = <String> ['one', 'two', 'three'];
num.retainWhere((item) => item.length == 3);
print(num); // [one, two]
```

30. setAll — start with index and overwrite values Set

E.X
```
final list = <String> ['a', 'b', 'c', 'd'];
list.setAll(1, ['bee', 'sea']);
print(list); // [a, bee, sea, d]
```

31 setRange — Nul. cuksis

32 shuffle — the elements of this list randomly

E.X
```
final num = <int> [1,2,3];
num.shuffle();
print(num); // [1,3,2] or some other random result.
```

33 Skip

E.X
```
final num = <int> [1,2,3,5,6,7];
final result = num.skip(4); // skip first 4 element
Print(result); // (6,7)
final skipall = num.skip(100); // ()
```

34 Sort — E.X o/p [-11, 0, 2, 13]

35 sublist — return new list, elements Between Start and End

E.X
```
final colors = <String> ['Red', 'green', 'blue', 'orange', 'pink'];
Print(colors.sublist(1,3)); // [green, blue]
Print(colors.sublist(3)); // [orange, pink]
```

36 take - take (length of) elements (start) with

```
final num = <int> [1, 2, 3];
final res = num.take(2); // [1,2]
final takeAll = num.take(100); //[1,2,3]
```

37 toList - convert dp in list format

38 toSet - " " " Set "

Now 2 Dart types of Exceptions

1. DefferedException - when a deffered library fails to load.
2. FormatException - when a string or some other data does not have an expected format
3. integerDivisionByZeroException - The number is divided by zero
4. IOException - its base class of I/O related Exception
5. IsolateSpawnException - isolated can not be created
6. Timeout - when scheduled timeout happens while waiting for an async result.
7. MissingPluginException - Plateform interaction failed to find a handling Plugin
8. NetworkImageLoadException - HTTP request to load a network image fails.
9. NullRejectionException - when the Promise is rejected with a null or undefined value
10. OSError
11. PathException
12. PlateformException
13. WebDriverException

3 ★ Try - The code block to run (to try logical coding part)

Catch - a code block to handle any Errors

finally - a code block to run regardless of the result. and print whatever Pass message.

Throw - generally must written in catch block

→ use to Explicitly raise an exception

→ Creating custom exception in the Dart.

Asynchronous programming in dart is characterized by the future and stream classes.

4 # asynchronous programming - async and await.

- fetching data over a network
- Writing to a database.
- Reading data from a file.

Key terms

1 Synchronous operation - It blocks other operations from executing until it completes.

2 Synchronous function - function only perform synchronous operations

3 asynchronous operation - Once, initiated, asynchronous operations allow other operations to execute before it completes.

4 asynchronous Fn - It perform at least one asynchronous operation and can be also perform synchronous operation

what is future

→ A future is a instance of the Future class.

→ A future represents the result of an asynchronous operation, and can have two state uncompleted or completed.

# Function	Type Signature	Descriptions
fetchLoginAmount()	Future<int> fetchLoginAmount()	Get the number of times a user has logged in

Ex of ~~Synchronous~~ Futures

```
Future <void> fetchUserOrder() {
    return Future.delayed (const Duration(seconds: 2), () =>
        Print(' Large Pizza');
    }

Void main () {
    fetchUserOrder ();
    Print (' Fetching user order...');
```

O/P
Fetching user order...
Large Pizza // print after 2 sec.

if i write
void main() async {
await fetchUser();
Then,
Print
large pizza
Fetching user order

Types of Parameters in Dart.

1. Positional Parameters

⇒ They are define by their position in the Fn Signature. Positional Parameters are enclosed with parentheses and Separated by commas.

Ex: void printDetails (String name, int age)

2. Named Parameters

⇒ identified by their name and enclosed with { }.

Ex: void printDetail ({String name, int age})

3. Optional Parameters

- Positional optional Parameters.
- Named optional Parameters

4. Required Parameters

5. Function Type Parameters. - ///(callback Fn);

Working with futures : async and await.

- To define an async fn, add async before the Fn body.
- The await keyword works only in async Fn.

Ex:

```
Future<void> printNumbersAsync() async {
    for(int i = 1; i <= 3; i++){
    print(i);
     await Future.delayed(Duration(seconds: 3));
    }}
void main() async{
    print('start');
  await printNumbersAsync();
    print('end'); }
```

O/P

```
start
1
// aft. 2 sec
2
// aft. 2 sec
3
End
```

Future Class Constructors

1. Future - creates a future containing the result of calling computation asynchronous with run
2. delayed - create a future that run its computation after a delay
   Ex
```
Future.delayed(const Duration(seconds: 1), () {
  Print('one second has passed'); // print after 1 second.
});
```
3. error
   Ex
```
Future<int> getFuture() {
  return Future.error(Exception('Issue'));
}

final error = await getFuture(); // throws.
```
4. microtask
5. sync - Returns a future containing the results of immediately calling computation.
   Ex
```
final result = await Future<int>.sync(() => 12);
```
6. value - Creates a future completed with value.
   Ex
```
Future<int> getFuture() {
  return Future<int>.value(2024);
}
final result = await getFuture();
```

Method

1. asStream - Creates a Stream containing the result of this future.
   - will produce single data or error event containing
   - if future never completes, stream will not produce any events.
2. catchError - Handles errors emitted (if any) by this Future.
   - This is the asynchronous equivalent of a "catch" block.

Ex

```
Future.delayed(
  const Duration(seconds: 1),
  () => throw 401,
).then((value) {
    throw 'Unreachable';
}).catchError((err) {
      print('Error: $err'); // Prints 401
  }, test: (error) {
      return error is int && error >= 400;
  });
```

3 then - Registers callbacks to be called when this future completes.

4 timeout - Stop waiting for this future after timeLimit has pass

5 Ex
```
result = await waitTask("Completed").timeout(const Duration(seconds:1),
    onTimeout: () => "timeout");
```

6 whenComplete - Registers a Fn to be called when this future completes.

Ex - This asynchronous equivalent of a "finally" block

```
void main() async {
  var value =
  await waitTask().whenComplete(() => print('do something here'));
  print(value);
}

Future<String> waitTask() {
  Future.delayed(const Duration(seconds: 5));
  return Future.value('done');
}
```

O/P
```
do something here
done
```

Stream

→ A Source of asynchronous data events

→ A ~~Stream~~ Stream Provides a way to recieve Sequence of Events
Each event is Either a data event, also call element of Stream.
error event, Someting has failed

→ You produce a Stream by Calling an async* Fn

→ There are 2 Kinds of Streams ① "Single-subscription" stream
② " broadcast " "

→ ① Single - Subscription

→ only a Single listener during the whole lifetime of the Stream.

→ twice on a single-subscription is Not allowed

→ It generally used for chunks of large contigous data like I/O.

② broadcast Stream

→ It allows any numbers of listeners, and fires it's events when they are ready.

→ It use for Independent events/observers.

→ defaut implementation of isBroadcast returns false

Methods

1. any - check wether test accepts any Element Provided by this stream
2. asBroadcastStream - Return a multi-subscription stream that Produces the same - event as this.
3. asyncExpand - Transform Each Elements in-to a Sequence of asynchronous events.
4. asyncMap - Create new Stream with each data event of this stream asynchronous mapped to a new event.
5. cast - Adapt this stream to be a Stream<R>, stream is curapped as Stream<A>
6. Contains - ex Compares each Elemt of this Stream to needle using

```
final res = await Stream.fromIterable(['year', 2021, 12, 24, 'Dart'])
    .contains('Dart');
print(res); // true
```

7. distinct - skip data event if they are equl to the previous data event

```
final Stream = Stream.fromIterable([2,6,6,8,12,8,8,2]).distinct();
stream.forEach(print); // 2,6,8,12,8,2
```

8 drain - Discards all data on this stream, but single when it is done or an error occured.

E.x final result = await Stream.fromIterable([1,2,3]).drain(100);
print(result); // 100.

9 elementAt - Return the value of the indexth data event of this Stream

10 every - Checks whether test accepts all elements provided by this Stream

11 expand - Transforms each element of this Stream into a sequence of elements

12 firstWhere - Finds the first element of this Stream matching test.

13 fold - Combines a sequence of values by repetetally applying combine

E.x final res = await Stream.fromIterable([2,6,10,8,2]).
.fold<int>(10, (Previous, element) => Previous + element);
print(result); // 38

14 forEach - Executes action on each element of this Stream.

15 handleError - Create a wrapper Stream that intercepts some error from this stream.

E.x
```
Stream.periodic(Const Duration(second : 1), (count) {
  if (count == 2) {
    throw Exception('Exceptional event');
  }
  return count;
}).take(4).handleError(print).forEach(print);
}
```

O/P
```
0
1
Exception: Exceptional event
3
4
```

16 join - Combines the string representation of elements into a single String.

17 lastWhere - Finds the last element in this stream matching test

18 listen - Adds a subscription to this stream.

19 map - Transforms each element of this stream into a new Stream event

20 pipe - Pipes the events of this stream into StreamConsumer

21 reduce - Combine a sequence of values by repetedly applying combine

Ex final res = await Stream.fromIterable([2, 6, 10, 8, 2])
.reduce((previous, element) => previous + element);
print(result); // 28.

22 SingleWhere - Finds the single element in this stream matching test.
23 skip - skip the first count data events from this stream.
24 skipWhile - skip data events from this stream while they are matched by test.
25 take - Provides at most the first count data events of this stream.
26 takeWhile - Forwards data events while test is successful.
27 timeout - Creates a new stream with the same events as this stream.
28 toList - Collects all elements of this stream in a List.
29 toSet - Collects the data of this stream in a Set.
30 transform - Applies StreamTransformer to this stream
31 where - Create a new stream from this stream that discards some elements.

Ex final Stream = Stream<int>.periodic(Const Duration(second: 1), (count) => count).take(10);
final CustomStream = Stream.where((event) => event > 3 && event <= 6);
customStream.listen(print);

o/p 4
5
6

Future and it's error handling

① then() - it's invoked on a Future that completes with an error
② catchError() - if we want to catch a specific error, of catch multiple Errors
③ whenComplete() - it is similar as finally block Print always !

Stream and it's error handling.

① .error
② handleError

Ex Future and Stream Error handling

```
Future<void> fetchUserData() async {
  await Future.delayed(Duration(seconds:2));
    // Intentionally throwing an error for demonstration
  throw Exception('failed to fetch user data');
}
void main() {
  ~~fetch~~ fetchedUserData()
      .then((_) {
        // Successful Completion
        print('user data fetched successfully');
      })
      .catchError((error) {
        // Error handling
        print('Error fetching user data : $error');
      });
  // Stream Example.
  Stream<int>.periodic(Duration(seconds: 1), (x) => x)
      .take(5)
      .listen((data) {
        print('Stream data: $data');
      }, onError: (error) {
        print('Stream error : $error');
      });
}
```

O/P

Stream data :0
Error fetching user data : Exception : Failed to fetch user data.

Stream data 1
Stream data: 2
" " 3
" " 4

Future Error handling

Ex

```
Future <void> fetchData () async {
    // Simulating fetching data asynchronously.
  await Future.delayed (Duration (seconds : 2));
  throw Exception ('Error fetching data');
}
Void main () {
  fetchData (). then ((value) {
     print ('Data fetched Successfully');
  }). catchError ((error) {
     Print ('Error : $ error');
  }). whenComplete (() {
     Print ('Task Complete, wether Successful or not');
  });
```

O/P

Error : Error fetching data.
Task complete, wether successful or not.

Extention method (Level 4)

→ add Functionality to Existing libraries
→ An Extention method is a Static method defined within a class that has a ~~recieve~~ receiver parameter. The receiver parameter is the obj. to which the method is apply.

Syntax

```
extention ExtentionName on ClassName {
  ReturnType methodName (ParameterType parameters) {
        // Implementation
  }
}
```

Static types and dynamic

→ You can't invoke extension method on variable of type dynamic

Ex ① dynamic d = '2';
Print(d.ParseInt()); // Runtime exception: NoSuchMethodError

Ex ② var v = '2';
Print(v.ParseInt()); // 2

Unnamed Extensions.

Extention on String {
bool get isBlank => trim().isEmpty;

⇒ You can invoke an unnamed Extention's static members only within the extention declaration

typedef

→ in Dart is we to create a user-define identity (alias) for a fn

How to use typedef

① Parameters of the fn
typedef function_name (parameters);

② Variable to a fn
typedef variable_name = function_name;

Ex variable and Collection

```
typedef List_Integer = List<int>;
void main() {
    List_Integer x = [1,2,3];
    Print(x);
}
```

o/p [1,2,3]

4 A mixin

→ reused in multiple class hierarchies.

→ TO use a mixin, use the 'with' keyword followed by one or more mixin name.

→ You can restrict a mixin's use by using the on keyword to specify the requried superclass

Ex

```
mixin LoggerMixin {
    void log(String message) {
        Print('Log : $message');
    }
}
Class Dog with LoggerMixin {
    String name;
    Dog(this.name);
    void bark() {
        Print('$name says Woof!'); }
void main() {
    var myDog = Dog('Buddy');
    myDog.bark();                    // Buddy say Woof!
    myDog.log('Having fun!');        // Log : Having fun!
```

6 A enums

→ enum are special kind of class
→ It we to represent a fixed number of constant values.
→ All enums automatically extend the Enum class.
→ They can't be subclassed, implemented, mixed in, instantiated

Ex
```
enum Gfg {
    Welcome, to, Flutter
}
void main() {
    for (Gfg i in Gfg.values) {
      print(i); }
}
```

o/p
```
Gfg.Welcome
Gfg.to
Gfg.Flutter
```

4 Lifecycle of Flutter App

1 resumed - app is in the foreground and is receiving user input.
2 inactivate - " " " " " but is not receiving user input
Ex This state can occur, when a phone call comes in or when user switches to other app.
3 Paused : app is in the background and is not visible to the user
Ex This state occurs, when the user presses the Home button or switches to another app.
4 detached - The app is not running at all, this state can occur
Ex when app is terminated by the OS. or when the device is restarted.
5 hidden - the app is completely hidden, minimize or observe another app

onResume - when a view in the application gains input focus.
onInactive - when the application loses input focus
onHide - when the application is hidden
onPause - when the application is paused
~~onDetach~~ onDetach - when an application has exited

★ Lifecycle of Stateful widget.

1 createState() - constructor is called, when initializes the widget, The createState method is then invoked, which creates the State obj for widget
2 initState() - this method is called when this obj is inserted in to the tree. We can initialize our variables, obj, streams, AnimationControllers, etc.
3 didChangeDependencies - this method call immediately after initState() when depends on some External data or changes via inherited widget.
4 build - ↓ Call every time the widget is rebuilt.
5 didUpdateWidget - when the widget is rebuilt with updated properties it's call after build(), and allow you to compare the privious and current widget properties
6 SetState - Triggers a rebuild of the widget when the State changes and the associated methods are called again
7 deactivate - method is call when the obj is removed from the tree. but might be inserted again
8 dispose - called when the widget is removed form the widget tree. Permanently, allowing you to release resources held by the widget.

★ Difference dependencies vs dev_dependencies

dependencies :

→ Packages your project needs to run
→ These are include in the app production build

Ex dio, get, intl, pinput

dev_dependencies

→ Packages you or a developer need during development
→ includes packages are required during development and testing but not necessary for the production deployment.
Ex flutter_test, flutter_lints, build_runner etc.

★ setState

⇒ used to notify the framework that the internal state of a widget has changed. and the widget UI needs to be rebuild.
→ setState does not replace the entire state of the widget, instead it only update the specified part of the state and triggers a rebuild of the widget's subtree.

→ When you call setState, flutter will re-run the build method of the widget, which allows you to update the UI based on the new state
→ It allows you to update UI in response to user input, network req, or other any event that changes the state of your app.
→ State Actually changes when you update it within the callback fn pass to the setState, ~~state~~
→ State happen in a controlled manner, allowing the framework to schedule a rebuild at the appropriate time

Note :- trimc) - method can be used to remove all the leading and trailing whitespace present in the String

classmate Date Page

Null Safety

⇒ ?, ?., ??, ??=, !

① ? → (Null check operator) It indicates with Null

E.x
```
void main () {
  int nonNullableVariable = 10; // Non-nullable variable
  int? nullableVariable = null; // Nullable variable
}
```

② ?. (null-aware access operator)
- Allow accessing properties or calling methods on potentially null objects without throwing an exception.

E.x
```
String? name;
var length = name?.length;
print (length); // null
```

③ ?? (Default Null-Aware operator)
→ Returns the left-hand side (LHS) if it's non-null, otherwise the right-hand side operator (RHS).
→ Useful for providing a default value if the (LHS) might be Null

E.x
```
var name;
var fullname = name ?? 'Yash shah';
print (fullname); // Yash shah
```

④ ??= (null aware assignment operator).
→ This operator assigns the value on its right to the variable on its left only if the variable on it's left is null.

Ex
```
String? name;
name ??= 'om';
print (name); // om
```

ASCII - A - 65
a - 97
zero - 0 - 48

(9) ! (null assertion operator)

→ This operator is used to assert that an Expression is non-null, if the Expression on its left is 'null', It throws runtime error

Ex

```
String? name;
Print(name!.length); // This will throw a runtime err
```

★ Dart Programs

★ Standard input in Dart

⇒ In Dart language, you can take Standard input from the user through the Console via the use of .readLineSync() Fn

→ To take input from the Console you need to import a library, named dart:io from libraries of dart

★ About Stdin Class:

⇒ This class allows the user to read data from standard input in both Synchronous and asynchronous way.

★ Standard output in dart

⇒ in dart, there are two ways to display output in Console.

① Using Print statement

② Using stdout.write() Statement

Note:- print() - brings the cursor to Nex Line
stdout.write() - don't bring the cursor to the next line, It remains in the Same line.

Stdin.readLineSync()
Scanf()
} - Both same take user Input

★ CodeUnits

→ The string property codeUnits in Dart programming language return the list of UTF-16 code of the characters of a string.

→ UTF-16 - (16-bit Unicode Transformation Format) is a character Encoding

Build modes 3 types

① ★ DEBUG

- The regular and most used mode
- Runs on all - Emulator, Simulator, Physical Device
- Here we solve any bugs/ error/design issue
- debugging tools are enabled - logs, Print statement, breakpoints and dev tools
- Hot reload, result instantly without having to restart the app
- The fast delivery is because of using JIT
  (Flutter doesn't recompile the whole app from scratch when updating)
- App is usually slow in debug mode. Performance is not optimized

② ★ Profile

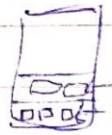
- Here, we care about performance of the app.
- You can analyse the memory management, Api or network calls any thing related to performance by using dev tools
- And we don't need debugging so some of debug tools are off but dev tool is on
- ex off tools. Printing, logs, breakpoints and hot reload.
- work only physical device no emulators
- Command flutter --profile

③ ★ RELEASE

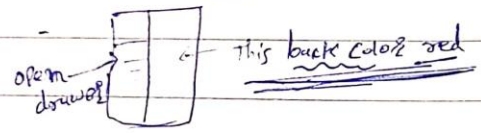
- Last step getting for Production.
- This how your app will look like on store.
- Fast start up, size and performance optimized both.
- Code is compiled to native in AOT.
- Command - flutter --release.

Widget

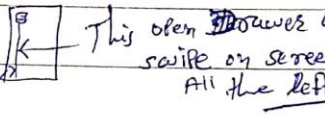
- Scaffold - persistentFooterButtons : [ElevatedButton —],
  └ This is a list of [textButton] widgets



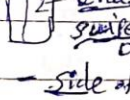
- bottomNavigationBar : BottomNavigationBar (
  items : [BNBItem(icon, label.)],
- backgroundColor
- resizeToAvoidBottomInset — use with keyboard overflow with scaffold
- extendBody : true // give result with appropriate — // if not body center text visible widely up sidely
- extendBodyBehindAppBar : true // if not body center text visible sidely down side
  body text center like this
- drawerScrimColor : red



- drawerEdgeDragWidth : 50,



  ↳ Horizontal swipe will open the Drawer
  Default value is 20.
- drawerEnableOpenDragGesture : false // doesn't use to horizontal swipe to open Drawer.
- endDrawerEnableOpenDragGesture : true // provide



- primary : false

  if true appbar size is little reduce (small)
  ↳ By default is true.
- persistentFooterAlignment : AlignmentDirectional.topCenter
  ↳ Default is centerEnd

- ShowBottomSheet — declare — ButtonPress → Scaffold.of(context).showBottomSheet
  - showDragHandle: true
  - backgroundColor
  - clipBehaviour
  - elevation
  - enableDrag : true
  - constraints : BoxConstraints(maxHeight: 500)

  sheetAnimationStyle
  (BuildContext context) {
  return Widget.

classmate

Date ______
Page ______

diff	showBottomSheet	showModelBottomSheet.
	① allow background click Event, like other Button, etc..	① don't allow, when click it close
	② PersistentBottomSheetController	② Future class type

★ showModelBottomSheet :- Button onpressed ma

- backgroundColor :
- ~~barrierColor~~
- barrierColor : // Set color back of smbsheet, also override the background of text other widget
- ~~constrain : BoxConstraint(maxHeight : 145) // To allow only reduce size, that mean's we pass the size low, to compare to SizeBox, Container Height // Not use to increase the smbsheet size.~~
- Constrain : BoxConstraint(maxHeight : 300) ;

  This mean, sizebox, container ni height declare કરી ને આપી. But smbsheet ની

suppose 500. (Half screen) → Default Height ને reduce થઈ શકે. increase ના થાય.

- isDismissible : true ~~Back~~ SMBS ની back પર click કરે તો close થાય

  False SMBS ની back પર " " " " નહીં થાય.

- isScrollControlled
- showDragHandle
- useRootNavigator
- useSafeArea
- sheetAnimationStyle

- Context : Context
- builder : (BuildContext context) {
  return

★ bottomNavigationBar : BottomNavigationBar(

- currentIndex
- SelectedItemColor
- onTap
- backgroundColor
- elevation
- enableFeedback
- iconSize
- mouseCursor
- selectedFontSize
- showSelectedLabels
- type
- unselectedIteamColor
- fixedColor

- selectedIconTheme
- unselectedIconTheme
- unselectedFontSize
- SelectedLabelStyle
- unselectedLabelStyle
- showUnselectedLabels
- landscapeLayout
- useLegacyColorScheme.
- required items : [
  BottomNavigationBarItem(
  required icon

★ Floating Action Button: - Scaffold properties.

- returntype Widget(itself)

FloatingActionButton(

- required onPressed
- tooltip
- foregroundColor
- backgroundColor
- focusColor
- hoverColor
- elevation
- min
- shape
- hoverElevation
- splashColor // Clickable Color.
- child

FAB Size types

FloatingActionButton.small
" .large
" .extended → - required onPressed and label

★ FloatingActionButtonLocation. endFloat is default

FABL. centerDocked

FABL. centerFloat — Not center - just above to BNB

FABL. CenterTOP

FABL. endDocked

FABL. endTop

FABL. startDocked

FABL. startFloat

FABL. startTop

FABL. miniCenterDocked

above All twice with mini

FABL. endContained - Touched

Center, Start, End — with Docked apply above BNB

- Float little above BNB

★ TabBar

- return DefaultTabController

required { child: length: 4 }

- TabBarView(

required children: []

Controller

- TabBar(

required list of tabs: []

type

- controller
- dividerColor
- dividerHeight
- indicatorColor
- Indicator
- isScrollable
- Padding
- indicatorPadding

TabBar.secondary

required List tabs: []

type

Tab()

★ SingleChildScrollView

- scrollDirection
- reverse
- Padding
- Physics
- Controller
- child
- KeyboardDismissBehaviour

* A GlobalKey is used here to identify the form and validate input. classmate
↳ [definition] → unique across the entire app.

Date
Page

* FormField. a single form fild widget that maintains the current state

* Form
required child
Key
canPop : false . it will not allow to back.
onPopInvoked
onChanged
autoValidateMode : AutovalidateMode. always, disable, onUserInteraction

* Difference Btn TextFormField and TextField

- If you making Form where you required save, reset, or validate operations use TextFormField.
- Else For simple user input capture TextField is sufficient.

TextField		TextFormField
decoration	onTap	- textCapitalization
Controller	onChanged	- textInputAction.next
obscureText	onSubmitted	- ~~Text~~ showCursor
contentPadding	cursorColor	- Validator
errorText	keyboardType	- AutovalidateMode
SuffixText		
SuffixIcon		
Prefix		
obscuringCharacter : '*';		
readonly		
TextDirection		
textAlign		
hintText		
icon		
lable		
fillColor		
border		

★ Expanded (
- required child
- Key
- flex

★ ColoredBox (
- required Color
- Child
- Key

★ Divider
- Color
- height
- thickness
- indent
- endIndent
- Key

o Ex
horizontal ①

★ VerticalDivider
Color
Width
thickness
indent
endIndent
Key

vertical. Vertical ①

★ Align (
- alignment - AlignmentDirectional.CenterStart;
- widthFactor
- heightFactor
- Child

★ SizedBox.expand
- Child
- Key

★ Card
- Color
- elevation
- Shape
- shadowColor
- borderOnForeground : true
- margin
- Child
- SemanticContainer : true default
- clipBehavior
- Key

★ Button

1 ElevatedButton
2 TextButton
3 OutlinedButton
4 FilledButton
5 FilledButton.tonal
6 TextButton.icon
7 ElevatedButton.icon
8 OutlinedButton.icon

Same All the properties like, ElevatedButton here, →

Dialogs - type Future

AlertDialog - for dialogs that have a message and some buttons.

SimpleDialog - for dialogs that offer a variety of options

ShowDialog - which actually displays the dialog and returns it's result.

① ElevatedButton

required onPressed, child

style

autofocus

clipBehavior

stateController

iconAlignment

focusNode

onLongPress

onHover

onFocusChange

⑤ A filled tonal button is an alternative middle ground between FilledButton and OutlineButton

⑦ ElevatedButton.icon

required onPressed, label

29-05-24

★ Dialog

- ① Alert . example message Cancle discard
- ② Simple . Ex set backup account
- ③ Conformation . Ex Phone ringtone set
- ④ Full screen . Ex Calender new event

- ○ ElevatedButton (onPressed:) showDialog (builder : (BuildContext context) => Dialog (
  - required context, builder
  - barrierDismissible : true
  - barrierColor
  - barrierLabel
  - useSafeArea
  - routeSetting
  - anchorPoint

  - backgroundColor
  - elevation
  - shadowColor
  - surfaceTintColor
  - shape
  - child
  - Alignment
  - clipBehavior
  - insetAnimationDuration
  - insetAnimationCurve

★ Dialog.fullscreen(
- child
- backgroundColor
- insetAnimationDuration
- insetAnimationCurve
- key

★ FadeInImage.assetNetwork(
required placeholder, image - Both type String
- Height
- Weight
- Fit
- Color
- Alignment

↓ Button actions
★ showAdaptiveDialog(builder:) => AlertDialog.adaptive()
required context, Builder,
- same dialog

★ AlertDialog
- icon
- iconPadding
- iconColor
- title
- titlePadding
- buttonPadding
- backgroundColor
- elevation
- shadowColor
- surfaceTintColor
- shape
- scrollable: true
- titleTextStyle
- content
- contentPadding
- contentTextStyle
- actions: []
- actionPadding
- actionAlignment
- semanticLabel
- insetPadding
- clipBehavior
- alignment
- key

- scrollController
- actionScrollController
- insetAnimationDuration
- insetAnimationCurve

★ Image
- Primary 2 way
① Displaying an image from Assets
② " " " from Network
- Assets = use for local image store within app assets
- network = " " images loaded from internet.

★ Image.assets(, - width
string - height
- fit
- alignment
- cacheHeight
- cacheWidth
- semanticLabel
- color
- opacity
- frameBuilder
- errorBuilder
- repeat
- Alignment

★ Image.network(,
src
- scale
- frameBuilder
- errorBuilder
- loadingBuilder
- headers
other same image asset

★ ClipBehavior - Clipping can be particularly useful for managing overflow, performance optimization, creating specific visual effect

★ cacheExtent - can enhance scrolling experience, preloading off-screen content

★ semanticChildCount - no of children will contribute semantic information (int)

★ FadeInImage.assetNetwork(
- required placeholder, image

★ ListView
- children []
- scrollDirection
- reverse
- controller
- shrinkWrap - allow widget or layout to adjust its size based on its content.
- physics - to control the behaviour of scrolling and other motion-based interaction within widgets, speed, boundaries
- itemExtent - impact in list height direct
  return type int
- cacheExtent
- semanticChildCount
- clipBehavior - what part widget visible or invisible when its boundaries are exceeded.

★ ListView.custom
required childrenDelegate

★ ListView.builder(
- required itemBuilder.
- scrollDirection
- reverse
- primary
- physics
- padding
- prototypeItem
- itemCount
- cacheExtent
- keyboardDismissBehavior
-

★ ListView.separated(
required itemBuilder
separatorBuilder
itemCount

30-05-24

★ PageView - whole page scroll ho.
- children: []
- controller
- scrollDirection: Axis.horizontal // Default
- reverse
- Physics
- PageSnapping: true // if false scroll list type ho
- onPageChange: (index) => Print('$ index');
  // mouse pass
- clipBehavior
- DragStartBehavior.start

★ PageView.builder(
- required itemBuilder
- itemCount
- scrollBehavior
- controller
- scrollDirection
-

★ PageView.custom(
required childrenDelegate(
required SliverChildBuilderDelegate -
widget type
builder:
childCount
int findChildIndexCallback
addAutomaticKeepAlive: true
addRepaintBoundaries:

P.V - allow you to swipe b/n 3 page

PVB. - dynamically build it's children using builder Fn

PVC. - That provide more control over the children using a custom 'SliverChildDelegate'.

★ SafeArea

- required child
- left
- top
- right
- bottom

} default true.

- minimum : inset around the content. =
  - EdgeInsets.zero
- maintainBottomViewPadding : false
  - Helps to avoid resizing content when the keyboard is display

★ Gridview - A scrollable, 2D array of widgets.

Ex: Mobile Galleries

- required { - itemBuilder
  - gridDelegate : SliverGridDelegateWithFixedCrossAxisCount(crossAxisCount : 2) (int type) (option after with 2 types.)
  - ② MaxCrossAxisExtent(maxCrossAxisExtent) (double)

num of column

- key
- scrollDirection
- reverse
- controller
- primary
- physics
- shrinkWrap
- padding
- itemCount
- clipBehavior
- addAutomaticKeepAlives
- addRepaintBoundaries
- addSemanticIndexes
- cacheExtent
- restorationId

Other properties

- mainAxisSpacing - Horizontal space b/n children
- crossAxisSpacing - Vertical " " "
- childAspectRatio - Aspect ratio of each child
- mainAxisExtent - height effect.

use to adjust the height of the items

★ Gridview.builder

- required { - gridDelegate
  - itemBuilder

★ Gridview.count

- required - int crossAxisCount
- childrean: []

★ Gridview.custom

- required { gridDelegate,
  - childrenDelegate : SliverChildBuilderDelegate((BuildContext, int) widget) (builder Fn)

GridView - create a grid with a fixed number of children

v. builder - more efficient for a large number of children as it lazily builds the grid tiles on demand

v. count - shorthand for creating grids with fixed number of tiles per row

v. custom - more customization of the grid layout, such as a specifying a custom sliverchildDelegate

v. extent - specifying the maximum extent of each tile, which makes the grid adaptive based on the available space.

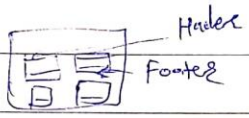
★ GridTile

required child

header -

footer -

key -



★ Wrap - works like row of Column, overflow widget ને Auto set કરે.

without column

- children: []
- direction: Axis.horizontal // default
- alignment: WrapAlignment.center
- spacing: 8.0 // horizontal default
- runSpacing: 8.0 // runtime space ob. // opposite દાણો.
- textDirection: textDirection.rtl // right to left point
- verticalDirection: verticalDirection.up // e.x 1 3 / 2 4 o/p 2 4 / 1 3

num of column

Horizontal space b/w children

vertical " " "

aspect ratio of each child

weight effect.

items

★ Stack -

- alt children []
- alignment: AlignmentDirection
- textDirection
- clipBehavior: Clip.none ① e.x  ② e.x Clip.AntiAlias 
- fit: StackFit.Expand
- key

★ Positioned

- required child
- height - left
- width - key
- top
- bottom
- right

★ List.generate - return type List<Widget>

- growable: true
- <length int>,
- (widget Function(int) generator) {
  ex (index) {
  return widget(
)
  }

materialApp - widget set up initial route and defines name routes.
Navigator - .. method like 'pushNamed' and 'popUntil' are used to navigate b/n Page.
onGenerateRoute - callable is used to handel dynamic generated routes.

Properties
~~Color : Colors.Primary~~
Color : Colors.primaries [index % Colors.primaries.length],
↳ 18 colors avail. 1 is 1 after then repeate all.

★ Navigator Properties
- MaterialApp(
- initialRoute : '/',
- routes : {
'/' : (context) => const HomePage(),
'/second' : (context) => secondPage(),
},
- onGenerateRoute : (setting) {
RouteFactory
if (setting.name == '/third') {
return materialPageRoute (builder: (context) => ThirdPage());
}
return null;

→ HomePage - onPress { Navigator.pushName (context, 'second');
Second " - " { " . " (" , '/third');
third " - " { Navigator.popUntil (context, modalRoute.withName('/'));

★ Navigator method : Psh
- Push : adds a new route to the top of the stack.
- Pop : Removes the top route from the stack.
- PushName : add new named route
- PopUntil : Pops route until a certain condition is met
- pushReplacement : Replace current route with new route
- canPop : checks if there is at least one active route in the stack.
- of : A static method to access the Navigator from the context.

* PushReplacement :- Replace the current route of the navigation disposing the previous route once the new route has allocate finished.

* PushAndRemoveUntil :- Push the given route onto the navigator that most tightly encloses the given context, and then remove all the previous routes until the predicate returns true.

★ Shared Preference

- Flutter are used for storing simple data in key-value Pairs
- Useful for saving user preferences and small amounts of data that need to persist across app session.
- Ex: User Preferences
  - Theme setting (dark mode or light mode)
  - Language localization setting

  User Data:
  - Username or user IDs
  - session tokens or authentication states

  App setting:
  - users can customize, like font size, or display options

- write data
- Read data

★ SwitchListTile

- required - value (bool type), onChanged (Fn bool)
  - activeColor
  - activeTrackColor
  - onFocusChange
  - title
  - subtitle
  - isThreeLine
  - dense
  - contentPadding
  - selected
  - shape
  - hoverColor
  - secondary (Ex. leading)
  - controlAffinity : ListTileControlAffinity.leading,

    ★ Swap the leading and trailing.

What is Statemanagement and why it is use?

→ It is manage the state, Ex widget ki action se change aaye.
↳ counter App + Button se text change - text a jaye

classmate Date Page

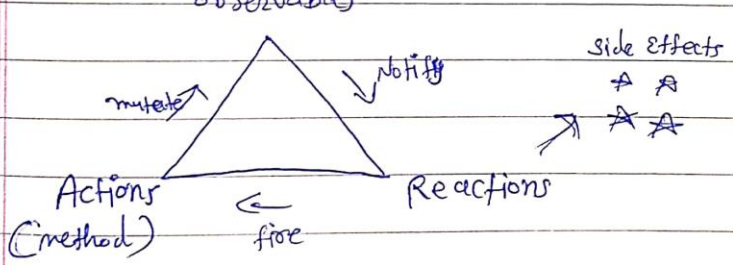
★ Statemanagement Technique

- MobX (2019)
- Getx (2019)
- Provider (2018)
- Block (2018)

★ MobX

- A popular library on observables and reactions

★ Core Concepts



★ At the heart of mobx use three important Concepts.

1. observables
2. Action
3. Reactions.

★ Getx

- 3 basic principles ① Productivity ② Performance and ③ organization
- Get has 2 diff State managers simple state manage (which we'll call GetBuilder, reactive State manage (Getx/Obx)

Ex var name = 'yash'.obs; // to make it observable

★ Obx(() => Text("${controller.name}")); // and in the Ui, when you want to show that value and update the screen whenever the value changes, simply do this

- GetMaterialApp(
- Get.to(NextScreen());
- Get.toNamed('/details'); // Navigate to new screen with name
- Get.back(); // To close snackbars, dialogs, bottomsheets, anything you would normally close with Navigator.pop(context);
- Get.off(NextScreen()); // To go to the next screen and no option to back to the previous screen (use in splashscreen, login screen, etc.)
- Get.offAll(NextScreen()); // To go to the next screen and cancle all previous routes (useful in shopping carts, polls, tests)

Dependency management

- Controller controller = Get.put(Controller());

Delete Data

```
// Remove data for the 'counter' key.
final success = await prefs.remove('counter');
```

Shared Preference

Write data

```
// obtain shared preferences.
final SharedPreferences prefs = await SharedPreference.getInstance();

// save an integer value to 'num' key
await prefs.setInt('num', 10);
// save an boolean value to 'flag' key.
await prefs.setBool('flag', true);
// Save an double value to 'decnum' key.
await prefs.setDouble('decnum', 1.5);
// save an String value to 'name' key.
await prefs.setString('name', 'Stust');
// save an list of strings to 'items' key.
await prefs.setStringList('items', <String>['Earth', 'moon', 'sun']);
```

Read Data

```
// obtain shared preferences.
final prefs = await SharedPreferences.getInstance();

// get an integer value from 'num' key.
await prefs.getInt('num');
// get an boolean value from 'flag' key..
await prefs.getBool('flag');
// get an double value from 'decnum' key.
await prefs.getDouble('decnum');
// get an string value from 'name' key.
await prefs.getString('name');
// get an list of strings from 'items' key.
await prefs.getStringList('items');
```

Path to store

/data/data/<your_package_name>/shared_prefs

AndroidManifest.xml file

Simple username store in shared preference

Step:

① class _HomePageState extend State<HomePage> {
TextEditingController _controller = TextEditingController();
String _savedValue = '';

② initState in super.initState();
_loadData();

③ Future<void> _saveData() async {
final prefs = await sharedPreferences.getInstance();
await prefs.setString('key', _controller.text);
setState(() {
_savedValue = _controller.text;
});
}

Write data

key, value (store data)

④

Future<void> _loadData() async {
final prefs = await sharedPreferences.getInstance();
setState(() {
_savedValue = prefs.getString('key') ?? 'no data';
});
}

Read data

Only key (get, read specific)

⑤

TextField(
controller: _controller

ElevatedButton(
onPressed: () {
_saveData();

Text('saved value: $_savedValue');

Library/Preferences/<your_bundle_identifier>.plist
is

★ Mobx

- in pubspec.yaml add (packages)
  - this 4 is required to use mobx statemanagement.
  - ① mobx : } dependencies
  - ② flutter_mobx : }
  - ③ mobx_codegen : } dev_dependencies
  - ④ build_runner }
- In terminal hit command -

① observables - Value that changes overtime and can notify any listeners of it's changes

② actions - Fn that can change your observable so an action. It can change value of one or more observables

★ @observables - observable is the variable, Uint, Map, Future, Stream

★ @action - declare with method increment.

★ @observer - update the user interface or perform other tasks, state change (reaction)

↳ required builder. widget function(BuildContext)? builder.

- String? name
- bool? warnWhenNoObservables
- Key

ex true

```
Observer(
  builder: (_) => Text(
```

↳ Fn produce a widget tree

★ Computed observables - what can be derived should be derived automatically.

- The state of your application consists of core-state and derived-state. The core state is state inherent to the domain you are dealing with for ex. Contact entity that firstName and LastName form the core state of Contact. However, fullName ~~from the core state~~ is derived state, obtain by combining firstname and lastname

ex part 'contact.g.dart';

★ @Computed

```
String get fullName => '$firstName, $lastName';
```

@readonly

Book ~~@observ~~ currentBook - this _currentbook as read only observables

@Computed - this properties as computed values that derive their values from other observables.

CMD :- flutter pub run build_runner build — get the store.g file

" " " " " watch

if you want to watch for changes and regenerate the code automatically.

getx

- Stateless {
  final CounterController controller = Get.put(CounterController());
- Obx(() {
  return Text(
  'Counter : ${controller.count}',
  );
- var count = 0.obs;
  RxInt count = 0.obs;
  List : obs
- controller.increment;

mobx

- counter_store.dart
- part 'counter_store.g.dart';
  class CounterStore = _CounterStore with _$CounterStore;
  abstract class _CounterStore with Store {
  @observable
  @action
  }
- counter_widget.dart
  final counter = CounterStore();
  class CounterWidget
  onPress : counter.increment
  CounterStore
- Counter_Store.g.dart

SHAIPASH

★ MobX

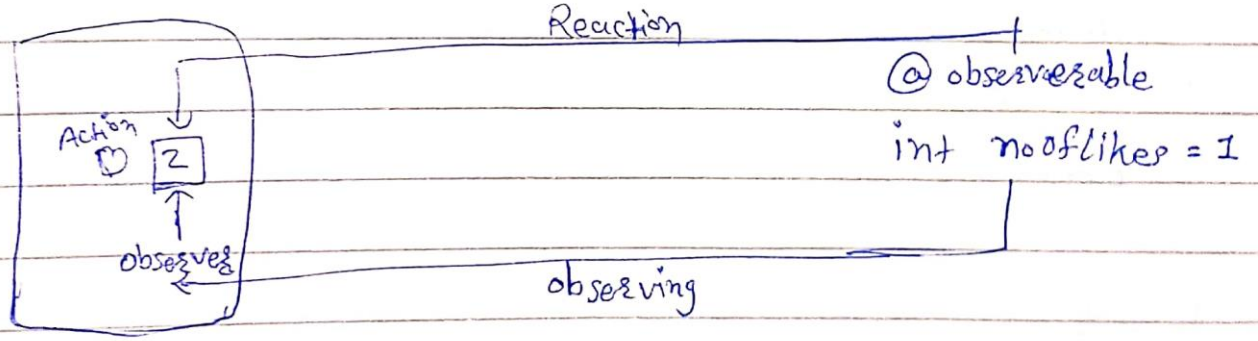
- works with Reactive Programming Concept & Asynchronous data stream

Observable - any data that can be changed.

Action - Any event that triggers the change in the value of observable.

Reaction - After an action and the change in observable, the action we take in return is called reaction

Ex Flow of MobX



★ Getx

It is mini framework within flutter framework, which basically combines improved version of different packages to make development easy and productive.

Getx provides

1. Routes Manager
2. State Manager
3. Dependency Manager
4. Extra Utilies - Get.to

① Route Manager

- Helps you avoid boilerplate code for Navigating in different screen

② State Manager

- It provides you a smart reactive state manager.
  - ① No need to work with streams.
  - ② Automatic disposal of states.
  - ③ No need of code generation.
  - ④ Simple and short code.

③ Dependency Manager.

- It Provides simple and easy dependency manager to inject dependency without need of Context.

Example of Provider dependency manager (Vs) getx dependency manager.

```
final service = Provider.of<ApiService>(context, listen: false);
final service = Get.find<ApiService>();
```

Web - Json to dart

classmate Date Page

Json

- Json stands for JavaScript object Notation
- It is a lightweight data - interchange format
- It is plain text written in JS. object notation
- It is used to send data b/n computers.
- It is language independent *
- It is very common task for apps that need to fetch data from the Internet.

Json Value

- In JSON, values must be one of the following data types:

→ string, number, object, array, boolean, null

→ Json data is written as Key value pairs.

Json object

- These {} represent json object which contains json value it could be string, integer, bool etc.

E.g. { "name" : "yash", "age" : 30, "car" : null };

Nested Json object

E.g.

Json obj.

```
{
  "status" : true,
  "mess" : "successfull",
  "data" : {
      "name" : "yash",
      "email" : "ye@gmail.com"
      "age" : "22",
  },
}
```

JSON ARRAY

E.g.

["Food", "BMW", "Key"]

E.g.

```
{
  "status" : true
  "data" : [
ARRAY   {
      "id" : 05,
    }
  ],
  "code" : 200
}
```

classmate
Date
Page

* And depending on how much JSON data you need to process, you have 2 option
  - (1) Write all the JSON Parsing code manually.
  - (2) automate the process with code generation.

* In flutter, to handel JSON serialization and deserialization Efficiently
  - Add in yaml file

```
dependencies :
    json_annotation :
dev_dependencies :
    build_runner :
    json_serializable :
```

* Create Model class

```
Part 'user.g.dart';

@JsonSerializable()
class User {
    name
    age
}
```

Command

```
flutter pub run build_runner build
```

☆ Animation

① TweenAnimationBuilder(
- required { tween : Tween<double>(begin: 0.0, end: 1.3),
  duration : Duration(
  builder : (context, size, widget)
- Key
- curve
- child
- onEnd.

② AnimatedContainer(
- required : duration
- ~~required~~ child
- curve
- alignment
- color
- height
- width

③ AnimatedAlign(
- required : alignment, duration
- child
- curve : Curve.fastOutSlowIn.
- heightFactor, widthFactor, Key.

creating smooth transition when changing the alignment of widget

④ AnimatedPadding(
- required : Padding, duration
- child
- curve
- key
- onEnd : Fn().

⑤ AnimatedBuilder
- It allows you to rebuild part of the widget tree when an animation change. It's commonly use to separate the animation logic from the widget tree.
- required animation, builder
- child

ex builder(context, widget, child){
return Transform.rotate(
- required angle :- double type
- child
- origin : offset
- alignment
- filterQuality.

⑥ AnimatedCrossFade ~~Container~~
- It creates a fade transition bt² two widgets when 1 widget is ~~replaced~~ replace by another
- required { firstChild (widget type)
  secondChild
  duration
  crossFadeState
- firstCurve
- secondCurve
- reverseDuration
- layoutBuilder

Date
Page

(7) AnimatedDefaultTextStyle

- This can be useful for creating smooth transitions b/t^n different text styles
- required { child, style, duration }
- TextAlign
- SoftWrap
- maxLine
- Curve

(8) AnimatedList

- It allows for dynamic updates to a list of items with animation
- It particularly useful when insert, remove, modify items while providing smooth animation for these changes
- required : itemBuilder . return Widget Fn (BuildContext, int, Animation<double>)
- initialItemCount = 0
- scrollDirection
- reverse
- Physics
- Padding, ClipBehavior, ShrinkWrap

(9) AnimatedOpacity

- automatically transaction the opacity of a child widget over a given duration.
- required Opacity type Double, duration
- curve
- child.

(10) AnimatedSize (

- required duration
- child
- Alignment
- Curve
- reverseDuration

with TickerProviderStateMixin

curve, easeIn

(11) FadeTransition - Blink effect Ex: Flutter Logo.
- required opacity
- child

(12) Hero
= It is used to create smooth transition b/h two screen. it typically used when navigating from one screen to another, where an element on the first screen animates to a corresponding element on the second screen. this create a visually b/h 2 screen
- required child, tag obj type

(13) PositionedTransition
- child
- rect : RelativeRectTween } required
animation that controls the child's size & position

(14) RotationTransition - Animates the rotation of widget
- required : turns rotation animation that controls the rotation of the child
- alignment, child, filterQuality.

(15) ScaleTransition - Ex Zoomin, Zoomout
- required scale - controll the scale of the child
- child, alignment, filterQuality.

(16) SizeTransition used 0 to 100 Left to right effect.
- Animates its own size and clips and aligns its child
required sizeFactor Animation
- child, axisAlignment, axis

(17) SlideTransition - Ex used Centers to Rightside to center Flutter Logo.
- required position type animation
- transformHitTests = true, child, textDirection.

SHAH FAISH

fit : stackfit. expand

11-06-24

* Constraints concept
- used to define how a ~~widget~~ widget can be sized and positioned within it's parent. generally describe the minimum and maximum ~~dimm~~ dimensions a child widget can take
- Constraint are essential for building responsive layouts.
- it we with ConstrainedBox widget

* ConstrainedBox(BoxConstraints type.
required Constraints :
child, key
* BoxConstraint(minWidth, maxWidth, minHeight, maxHeight. type double

* Dismissible - A widget that can be dismissed by dragging in the indicated direction
- required key, child
- background, direction, movementDuration, resizeDuration, secondaryBackground confirmDismiss, onResize, onDismissed, onUpdate, behavior, crossAxisEndOffset

* Opacity - A widget that makes it's partially transperent. like blure
- value B/w 0 to 1 double (whole)
- required Opacity. double
- child, key, alwaysIncludeSemantics.

* How to Apply Blur effect with widget?
→ BackdropFilter - A widget that applies a filter to the existing painted content and then paint child
- required : filter : ImageFilter.blur(sigmaX: 9.0, sigmaY: 11.0)
- child
- blendMode
  - compose
  - dilate
  - erode
  - matrix

Expanded | Flexible

FlexFit fit = FlexFit.loose,

Note ⇒ in column i have 3 containers 1 can height 200, 2 is Flexible with 200 height, 3 Expanded height 100 the ③ container not occupie what space BCoz before Flexible height that only

Code in

* Flexible
- A widget that controls how a child of Row, Column, or Flex flexes.
- required child
- flex int
- fit = FlexFit.loose (it take height of child)
- Key

fit = FlexFit.tight — same as Expanded

* Expanded
- A widget that Expands a child of Row, Column, or Flex so that the child fills the available space
- required child
- flex
- Key

* Spacer (flex, Key
- Spacer creates an adjustable, empty spacer that can be used to the spacing Btn widgets in a Flex container, like Row or Column
- flex-factor to use in determining how much space to take up.

* FittedBox(- Scales and positions it's child within itself according to fit
- child, clipBehavior, fit, alignment, key.

Ex fit : BoxFit. contain
- cover
- fill
- fitHeight
- fitWidth
- none - same actual size.
- scaleDown
- values = List <BoxFit>

★ GestureDetector
- A widget that detects gestures.
- child
- onTap, onTapDown, onTapUp,
- onTapCancel, onSecondaryTap,
- onDoubleTap
- onLongPress, onLongPressDown, start, cancle, UP, End
- onHorizontalDragStart, Update, End, Cancle
- onVerticalDragStart, Down, Update, End, Cancle.
- onPanDown, start, Update, End, cancle
- onScaleStart, Update, end,

Tap
Double Tap
Long Press
Vertical drag
Horizontal drag
Pan
Scale

★ Inherited Widgets
- Widget tree
- Element tree
- RenderObject tree

Diffesonce

	Inkwell	GestureDetector
Visual Feedback (ripple effect)	✓	✗
Complexity (more control)	✗	✓
	built in material widget	Not specifically tied to material design.

★ What is Rest API in Flutter ?
- REST API uses simple HTTP calls to communicate with JSON data.

definition - REST (Representational State Transfer) is an architectural style for designing Networked applications.
- It relies on stateless, client-server, cacheable Communication protocol
- The HTTP Restful application use HTTP requests to perform CRUD operations.

★ Key Components of REST API:

1. Endpoint/Resource :- A URL that points to a resource.

2. HTTP methods :-

What is API.

→ API (Application Program interface)
→ Allow communication Bt^n client and servers.

- GET : Retrive data from the server
- POST : Create new data and Post the server
- PUT/PATCH : Update existing data " "
- DELETE : Remove data. " "

3 Headers :- Carry metadata for request/response - Barrier, token, work API
Status Code - Range for

4 Body : Contains the data sent to the server (usually in POST and PUT req.)

Authentication	Authorization
In the Authentication process, the identify of users are checked for providing the access to the system	In authorization procer, ea the person's or user's authorities are checked for accessing the resources.
Users or person are Verified.	user or person are Validated
transmit info. through an ID Token	transmit information through an Access Token

Bearer Token

→ access token for the Authentication
→ A bearer token is security token that ~~generates~~ grants the bearer access to protected resources.
→ The B.T. is typically generated by an authentication server and is included in the HTTP Authorization header of request to the server that hosts the protected resources.

Status Code

- Alphanumeric String (Complex with additional info)
- Authorization header using "Bearer" scheme
- more secure, Supports token expiration

1 Informational responses
- 100 continue :
- 101 switching protocols
- 102 processing

- Success responses
  - 200 ok - req was successful and Server returned the req. data
  - 201 created - " " " " Server created a new resource
  - 204 No Content - " " " " Server did not return any data

- redirection responses
  - 304 Not Modified

- Client error responses
  - 400 Bad Request - The req. was malformed or invalid
  - 401 Unauthorized - It is not authorized to access the req. resource
  - 403 Forbidden - Client is authenticated but not authorized to access the req. resources
  - 404 Not Found - the req. resource was not found on the Server

- Server error responses
  - 500 internal Server error:
  - 502 Bad Gateway: recived an invalid response from an Upstream Server
  - 503 Service Unavailable: Server is temporarily unable to handle req. increased traffic or the server is undergoing maintenance

FutureBuilder

→ A Widget that builds itself based on the latest snapshot of interaction with a Future.