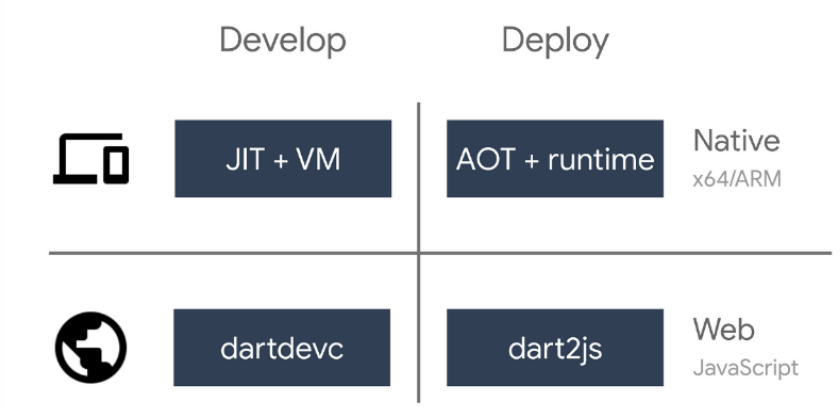
Google's Flutter

Version 3.0

index

Flutter, Dart background

- Dart, 최악의 언어?
  - 2011년 java script를 대체하기 위해 탄생
    - High learning curve & 대체가능한 언어존재(typeScript)
  - JIT(Just in Time), AOT(ahead of time) 컴파일 모두 지원
    - 개발중에는 JIT(hot reload), 출시할때는 AOT 컴파일러 사용
  - 2022.7월 현재 v 3.0.4
  - 50만개 이상의 앱이 Flutter로 개발됨

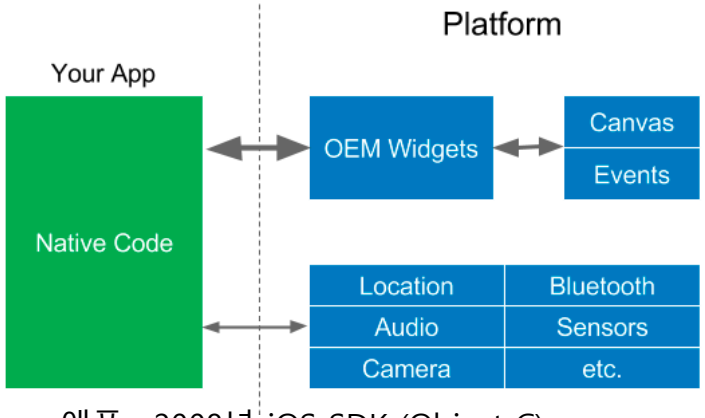


- Flutter
  - 2017.5월 출시
  - 2018.12월 v1.0
  - 2021.3월 v2.0
  - 2022.2월 v2.10, windows 지원
  - 2022.5월 v3.0, Linux, Mac 지원
    - Flutter SDK release : stable / beta/ dev/ master

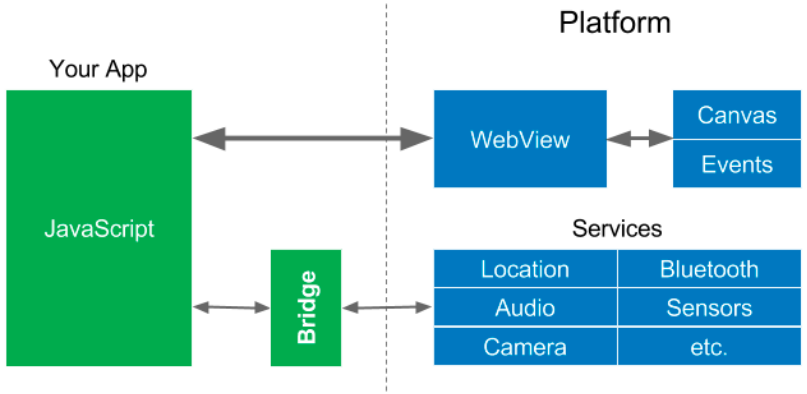
- Flutter 1.21부터는 Flutter SDK 설치시 Dart SDK 자동설치
- Cross platform : Android, iOS, Linux, Mac, Windows, Google Fuchsia, Web (-> Fuchsia : Google's new OS)

- 공식사이트 :
  - Dart : https://dart.dev
  - Flutter : https://flutter.dev
  - 무료 온라인 도서 : https://www.raywenderlich.com/books/flutter-apprentice

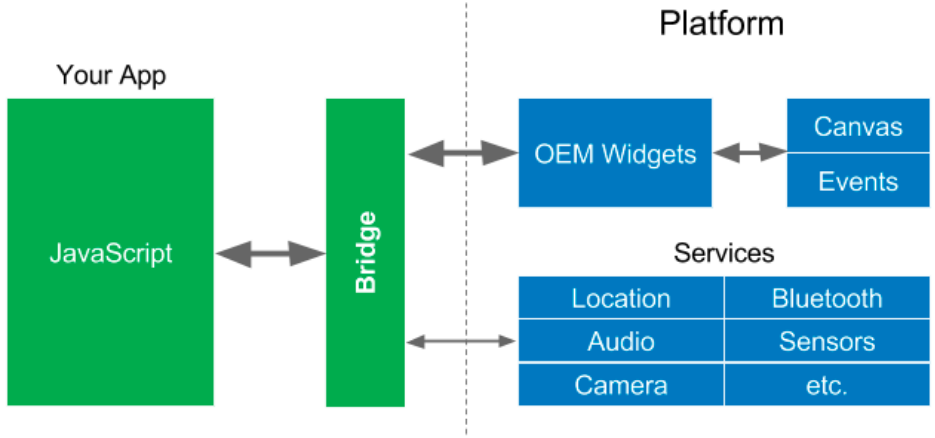
Flutter features 1/3



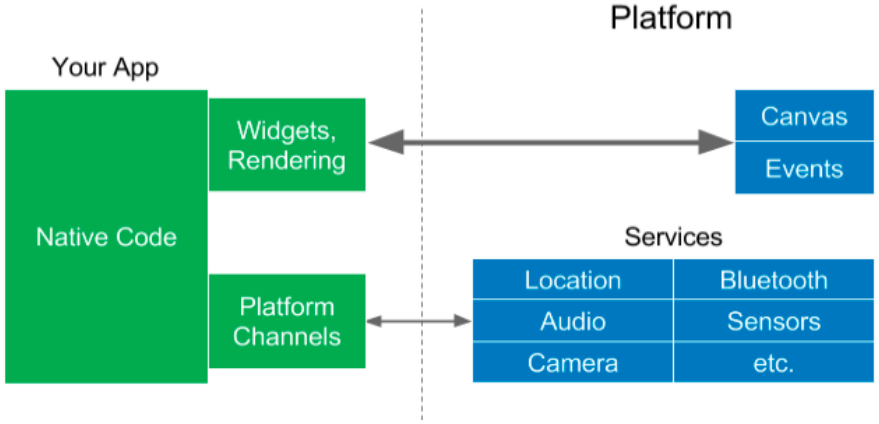
- 애플 : 2008년 iOS SDK (Object-C)
- 구글 : 2009년 Android SDK (JAVA)
- 각 플렛폼별로 독립적인 앱을 개발



- 크로스플렛폼 프레임워크시작 : javascript, WebView(HTML)
- 초기에는 복잡한 기능이 없어서 성능이 문제안됨



- 2015년 React Native : HTML, javascript
- 문맥교환 브릿지를 거쳐야하므로 성능 저하문제가 나타남



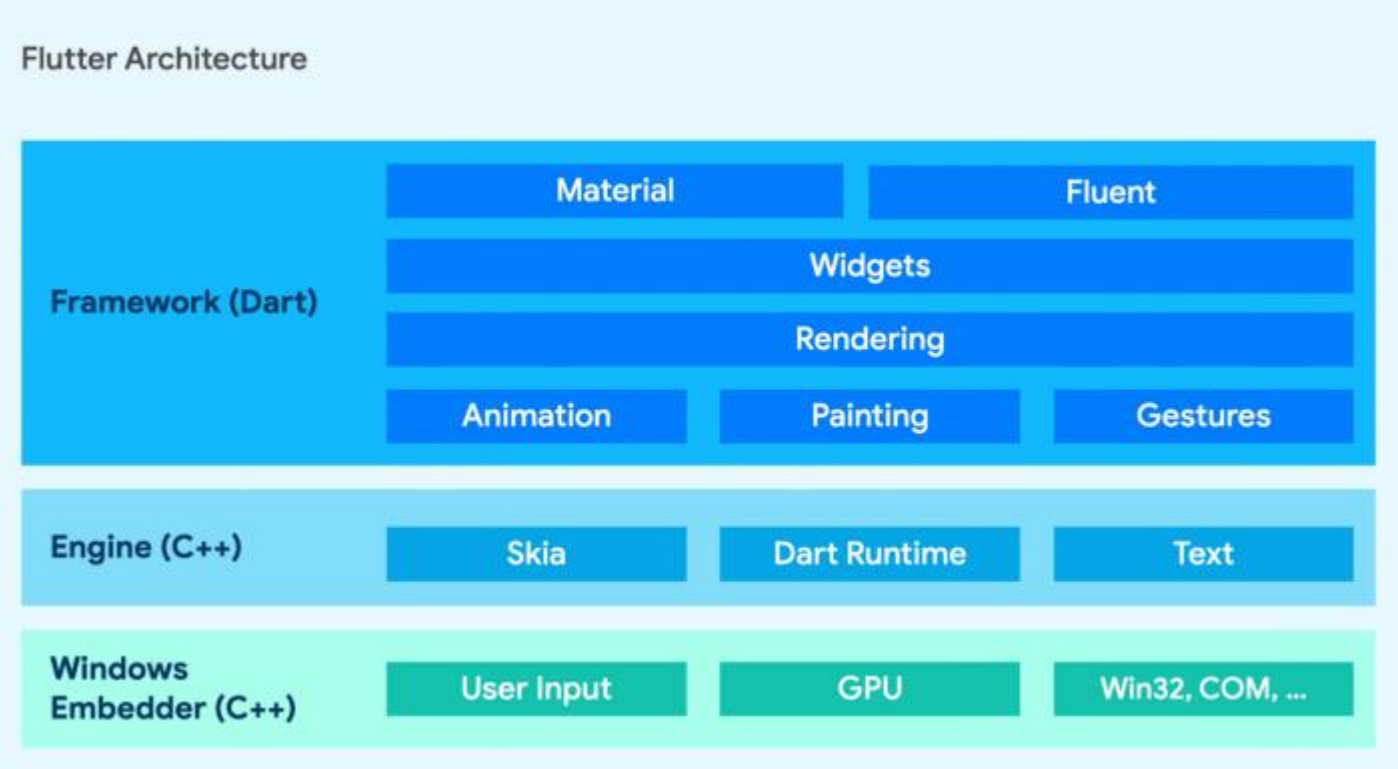
- Flutter : Dart컴파일 언어를 이용해 브릿지 성능 저하문제를 해결
- OEM위젯을 사용하지 않고 자체위젯을 사용
- 플렛폼과 직접 커뮤니케이션

Flutter features 2/3

- 특징
  - Cross-Platform APP Development Framework
    - 하나의 코딩으로 여러 OS 응용프로그램 개발
    - Android, iOS, WEB, Windows, Linux, Mac
  - Dart 언어기반
    - 깊이 들어가면 native 코드가 필요하므로 Kotlin, Swift도 사용가능
  - 통합개발환경 지원 : Android Studio, VS code ...
  - 빠른 개발(컴파일), hot-reload
  - 성능문제 해결 : 기존의 앱개발 도구에 비해 빠른성능
  - Material Design 제공 : 플렛폼에 따라 각각의 디자인 가이드에 맞게끔 화면 표현 (iOS 앱을 개발 할 경우를 위해 Cupertino 위젯제공)

- 단점
  - 웨어러블 디바이스앱 개발어려움
  - 지원되는 플러그인이 부족
  - 아직까진 국내에 개발관련 자료가 많이 부족

Flutter features 2/3



참고 https://developers-kr.googleblog.com/2022/03/announcing-flutter-for-windows.html

Road Map

- 2022년 중점전략 : https://github.com/flutter/flutter/wiki/Roadmap
  - 사용자 경험 : 사용자가 좋아하는 SDK를 제작하겠다!
  - 데스크탑지원 : 2022.2.3 stable 채널 제공. Windows, linux, mac OS 순으로 지원예정
  - Web : 성능개선, 플러그인 수준향상, 다양한 브라우저 지원, Flutter 가아닌 HTML 문서에 flutter application을 임베드 할 수 있도록.. 등
  - Material design 3 지원
    - Material design : flat 디자인의 장점을 살리면서도 빛에 따른 종이의 그림자 효과를 이용하여 입체감을 살리는 디자인 방식(2014 구글이 안드로이드 폰에 적용하면서 보편화됨) : https://material.io/design/introduction
    - Material you : 기존 머트리얼 디자인에서 한발 더나간 개념. UI요소들의 개인화가 강화됨 안드로이드 12의 기본 UI
    - iOS 스타일 = Cupertino 스타일
  - Dart 언어 개선

관련사이트, 앱

- Flutter.dev
- Dart.dev
- Material.io
- https://io.google/2022/products/flutter/intl/ko/
- https://www.facebook.com/groups/flutterkorea
- Holix 의 클럽 "Flutter와 Dart가 궁금하신가요?"

개발환경 구축

flutter.dev

Android Studio : 폴더구조

- android / ios 폴더 : 자동생성, native code 적용
- Lib : 소스구성하는 폴더, 파일
- Assets 은 앱과 함께 배포되는 정적데이터 파일
  - 이미지, 아이콘, json파일
  - Pubspec.yaml 파일에서 정의
- pubspec.yaml : 앱이름, 버전정보, 외부패키지 정의

```
flutter:
  assets:
    // 특정파일 포함
    - assets/my_icon.png
    - assets/background.png
    // 특정폴더 포함 : 폴더안의 모든 파일 자동으로 포함. 폴더의 하위 폴더는 각각 명시해줘야 포함됨
    - directory/
    - directory/subdirectory/
```

Android Studio : 중요메뉴

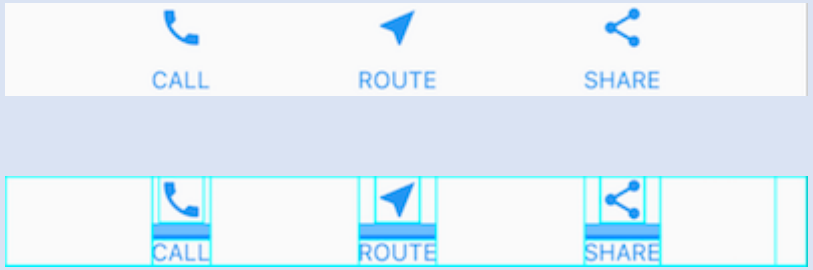
- Flutter SDK 위치,버전 : File > setting > Languages & Frameworks > Flutter
- Dart SDK 위치,버전 : File > setting > Languages & Frameworks > Dart
- Android SDK 위치,버전 : File > setting > Appearance & Behavior > System settings > android SDK
- 화면 폰트 설정 : File > setting > Appearance & Behavior > Appearance
- 화면폰트 사이즈 변경 : File > setting > Editor > General > change font size with Ctrl+mouse wheel
- Plugins : Dart, Flutter 모두 플러그인 형태로 제공
- Emulator 구동 : Tools > AVD manager
- Terminal, run... 창 : dock, float
- Terminal 창 : windows의 명령프롬프트
- Play, stop, hot reload, debug...

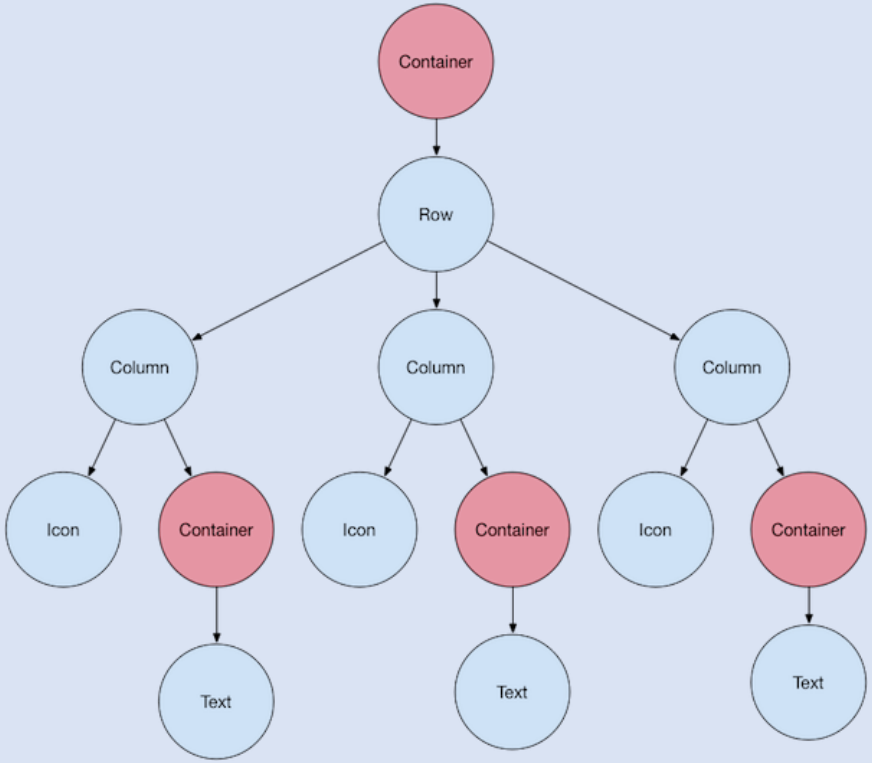
Android Studio : Short-key

- Ctrl+Shift+I : 위젯기본정보-파라메터
- Alt+Enter : 다른위젯 적용 : 교체, 감싸기, 삭제, 위치변경 등
- Ctrl+Shift+"+/-" : 위젯확장/ 축소
- Ctrl+F : 찾기(현재파일)
- Ctrl+Shift+F : 찾기(전체파일)
- Ctrl+R : 바꾸기(현재파일)
- Ctrl+Shift+R : 바꾸기(전체파일)
- Ctrl+Alt+L : 화면 정렬 : 콤마(,) 기준
- "st" 입력 : stateless/ statefullness 자동완성
- Ctrl+B : 위젯의 소스코드보기
- Ctrl+W : 위젯선택, 상위 위젯선택
- **Ctrl+Alt+O : 안쓰는 import 삭제**
- Flutter outline : 계층구조 시각화(우측면탭). 마우스우측클릭->위젯분리

Flutter basic : Layouts 1/3

- Flutter layout의 핵심 메커니즘 **Widget**
  - 거의 모든것이 위젯 :이미지, 아이콘, 텍스트 등 보여지는 모든 것이 위젯. 심지어 레이아웃도 위젯
  - 보여지지 않는 것들도 위젯 : rows, columns, grids ...





Flutter basic : Layouts 2/3

- 위젯은 UI를 빌드하는 데 사용되는 클래스.
- 위젯은 레이아웃과 UI 요소 모두에 사용.
- 간단한 위젯을 구성하고 복잡한 위젯으로 확대.
- 보이는 위젯 몇가지 -> 1
  - Text
  - Image
  - Icon

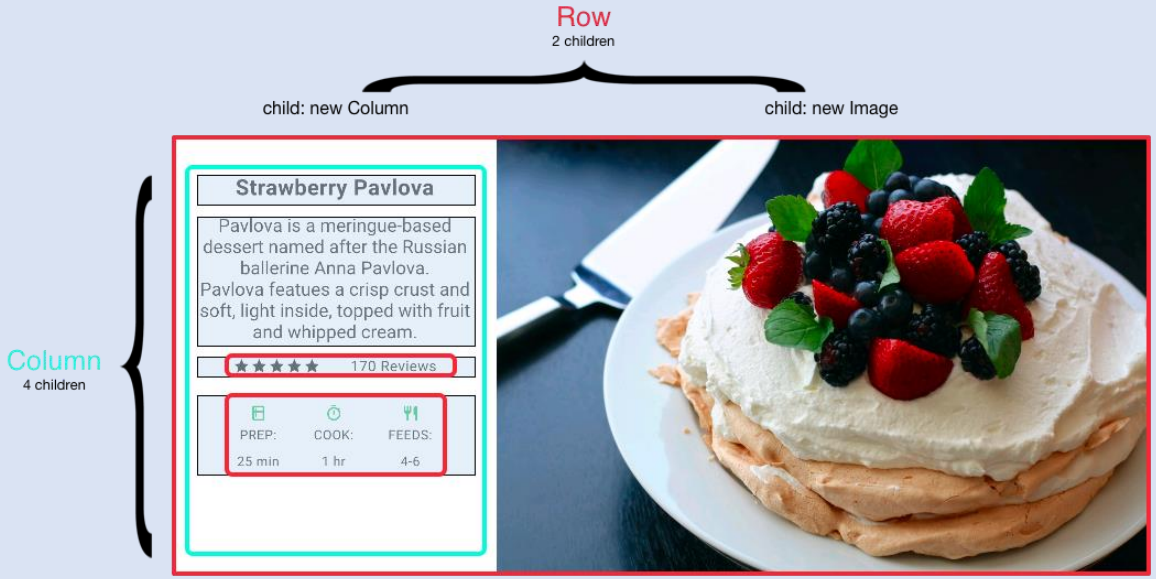
- 레이아웃 위젯에 보이는 위젯추가 -> 2
  - Center
  - Container

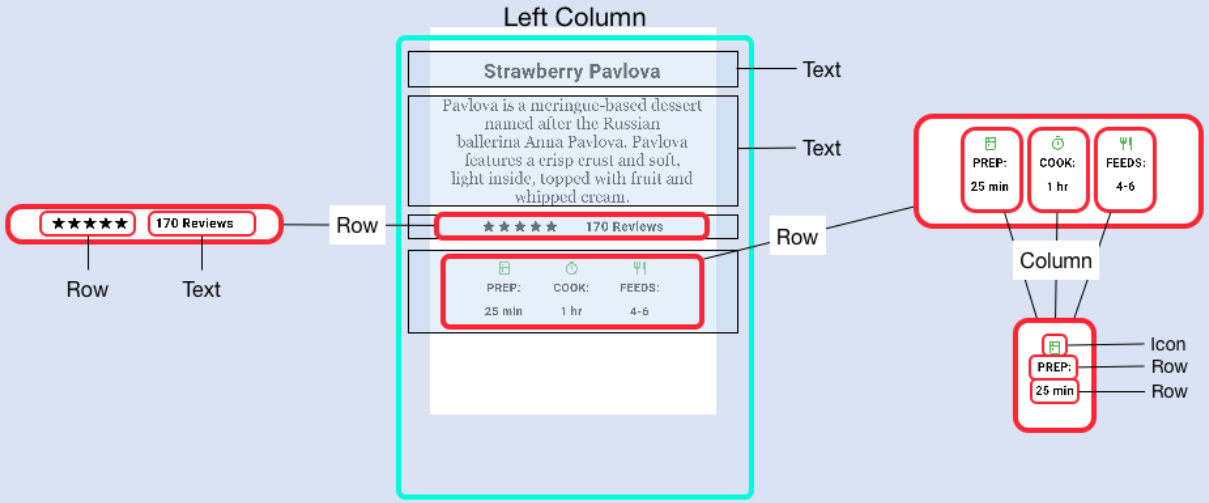
- 페이지에 레이아웃 위젯추가 -> 3
  - Material widget
    - Material Design 을 기반을 하는 다양한 기능이 라이브러리화 되어있다
  - Material Design :
    - 위키백과 : 플랫 디자인의 장점을 살리면서도 빛에 따른 종이의 그림자 효과를 이용하여 입체감을 살리는 디자인 방식을 말한다. 2014년 구글이 안드로이드 스마트폰에 적용하면서 널리 퍼지기 시작했다. 플랫 디자인과 마찬가지로 최소한의 요소만을 사용하여 대상의 본질을 표현하는 디자인 기법인 미니멀리즘(minimalism)을 추구한다.
    - https://material.io/

```
// 1
Text('Hello World'),

Image.asset(
  'images/lake.jpg',
  fit: BoxFit.cover,
),

Icon(
  Icons.star,
  color: Colors.red[500],
),

// 2
Center(
  child: Text('Hello World'),
),

// 3
class MyApp extends StatelessWidget {
  MyApp({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter layout demo',
      home: Scaffold(
        appBar: AppBar(
          title: Text('Flutter layout demo'),
        ),
        body: Center(
          child: Text('Hello World'),
        ),
      ),
    );
  }
}
```

Flutter basic : Layouts 3/3

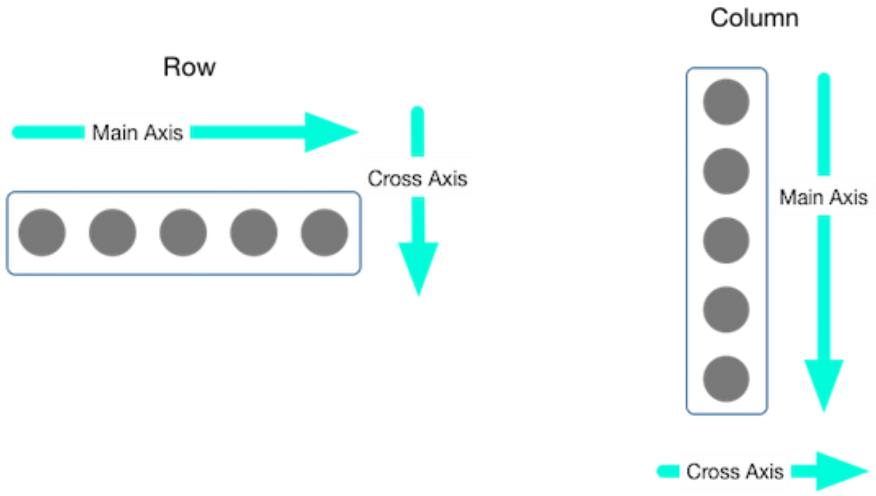
- 수직과 수평 레이아웃을 구성하는 위젯들
  - Row
  - Column





Flutter basic : Layouts : Aligning widget

- Row 또는 column의 정렬을 돕는 속성 두가지
  - mainAxisAlignment
  - crossAxisAlignment

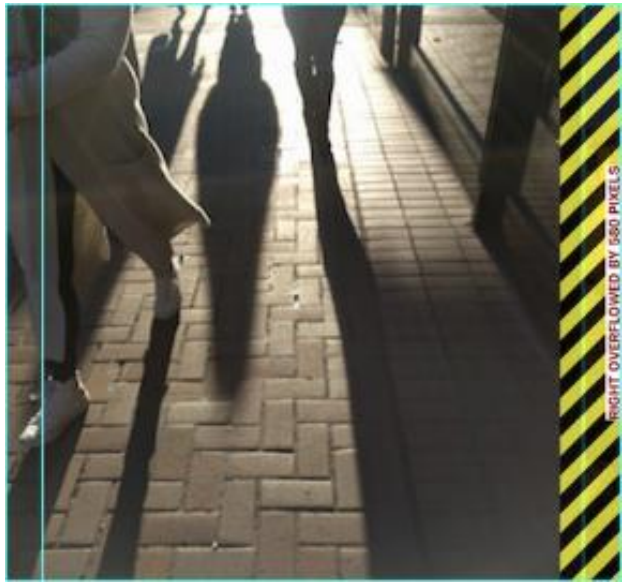



```
Row(
  mainAxisAlignment: MainAxisAlignment.spaceEvenly,
  children: [
    Image.asset('images/pic1.jpg'),
    Image.asset('images/pic2.jpg'),
    Image.asset('images/pic3.jpg'),
  ],
);
```

```
Column(
  mainAxisAlignment: MainAxisAlignment.spaceEvenly,
  children: [
    Image.asset('images/pic1.jpg'),
    Image.asset('images/pic2.jpg'),
    Image.asset('images/pic3.jpg'),
  ],
);
```

Flutter basic : Layouts : Sizing widgets

- 레이아웃이 장치에 비해 클경우 overflow 발생
- Expanded위젯으로 해결

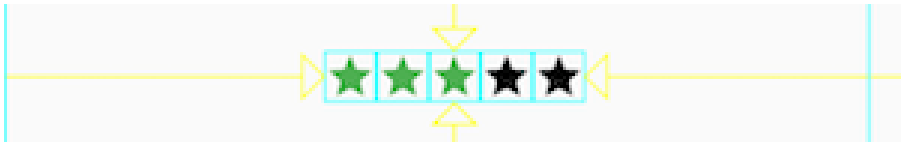



```
Row(
  crossAxisAlignment: CrossAxisAlignment.center,
  children: [
    Expanded(
      child: Image.asset('images/pic1.jpg'),
    ),
    Expanded(
      child: Image.asset('images/pic2.jpg'),
    ),
    Expanded(
      child: Image.asset('images/pic3.jpg'),
    ),
  ],
);
```

```
Row(
  crossAxisAlignment: CrossAxisAlignment.center,
  children: [
    Expanded(
      flex: 1, // default
      child: Image.asset('images/pic1.jpg'),
    ),
    Expanded(
      flex: 2,
      child: Image.asset('images/pic2.jpg'),
    ),
    Expanded(
      child: Image.asset('images/pic3.jpg'),
    ),
  ],
);
```

Flutter basic : Layouts : Packing widgets

- 기본적으로 Row, Column 위젯은 주방향으로 최대의 공간을 차지
- Children 항목에 위젯들을 연이어 배치하고 주방향의 사이즈를 최소화해주면 적당한 크기로 보여준다



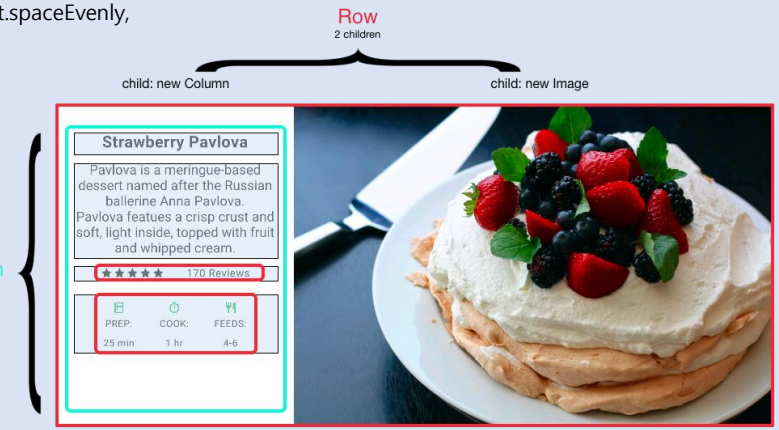
```
Row(
  mainAxisSize: MainAxisSize.min,
  children: [
    Icon(Icons.star, color: Colors.green[500]),
    Icon(Icons.star, color: Colors.green[500]),
    Icon(Icons.star, color: Colors.green[500]),
    const Icon(Icons.star, color: Colors.black),
    const Icon(Icons.star, color: Colors.black),
  ],
)
```

Flutter basic : Layouts : 복잡한 구성은 분리해서 코딩

- 화면이 여러 위젯의 복잡한 구조로 구성되어 있다면 적당한 부분을 구분 하여 별도의 코드로 구성

```
const descTextStyle = TextStyle(
  color: Colors.black,
  fontWeight: FontWeight.w800,
  fontFamily: 'Roboto',
  letterSpacing: 0.5,
  fontSize: 18,
  height: 2,
);

final iconList = DefaultTextStyle.merge(
  style: descTextStyle,
  child: Container(
    padding: const EdgeInsets.all(20),
    child: Row(
      mainAxisAlignment: MainAxisAlignment.spaceEvenly,
      children: [
        Column(
          children: [
            Icon(Icons.kitchen, color: Colors.green[500]),
            const Text('PREP:'),
            const Text('25 min'),
          ],
        ),
        Column(
          children: [
            Icon(Icons.timer, color: Colors.green[500]),
            const Text('COOK:'),
            const Text('1 hr'),
          ],
        ),
        Column(
          children: [
            Icon(Icons.restaurant, color: Colors.green[500]),
            const Text('FEEDS:'),
            const Text('4-6'),
          ],
        ),
      ],
    ),
  ),
);
```

```
var stars = Row(
  mainAxisSize: MainAxisSize.min,
  children: [
    Icon(Icons.star, color: Colors.green[500]),
    Icon(Icons.star, color: Colors.green[500]),
    Icon(Icons.star, color: Colors.green[500]),
    const Icon(Icons.star, color: Colors.black),
    const Icon(Icons.star, color: Colors.black),
  ],
);

final ratings = Container(
  padding: const EdgeInsets.all(20),
  child: Row(
    mainAxisAlignment: MainAxisAlignment.spaceEvenly,
    children: [
      stars,
      const Text(
        '170 Reviews',
        style: TextStyle(
          color: Colors.black,
          fontWeight: FontWeight.w800,
          fontFamily: 'Roboto',
          letterSpacing: 0.5,
          fontSize: 20,
        ),
      ),
    ],
  ),
);
```



>> https://docs.flutter.dev/development/ui/layout/tutorial

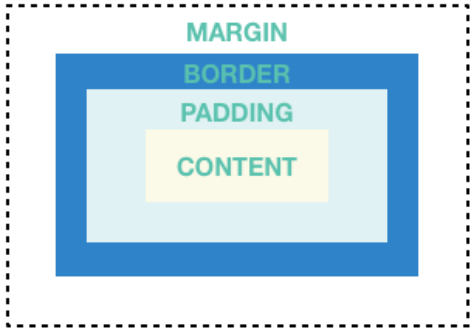
Flutter basic : Common layout widgets

- 자주 사용되는 대표적인 레이아웃 위젯
  - Standard widgets
    - Container
    - GridView
    - ListView
    - Stack
  - Material widgets
    - Card
    - ListTile

>> https://docs.flutter.dev/development/ui/layout

Flutter basic : Common layout widgets : Container()

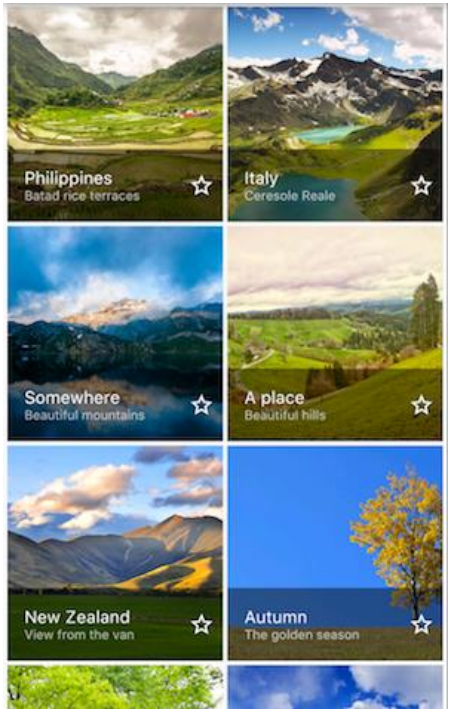
- 패딩, 마진, 경계선
- 배경색, 배경이미지
- 한개의 자식위젯이 가능 : Row, Column 위젯을 사용하면 복수의 자식위젯 사용가능



```
Widget _buildImageColumn() {
  return Container(
    decoration: BoxDecoration(
      color: Colors.black26,
    ),
    child: Column(
      children: [
        _buildImageRow(1),
        _buildImageRow(3),
      ],
    ),
  );
}

Widget _buildDecoratedImage(int imageIndex) => Expanded(
      child: Container(
        decoration: BoxDecoration(
          border: Border.all(width: 10, color: Colors.black38),
          borderRadius: BorderRadius.all(Radius.circular(8)),
        ),
        margin: EdgeInsets.all(4),
        child: Image.asset('images/pic$imageIndex.jpg'),
      ),
    );

Widget _buildImageRow(int imageIndex) => Row(
      children: [
        _buildDecoratedImage(imageIndex),
        _buildDecoratedImage(imageIndex + 1),
      ],
    );
```

Flutter basic : Common layout widgets : GridView

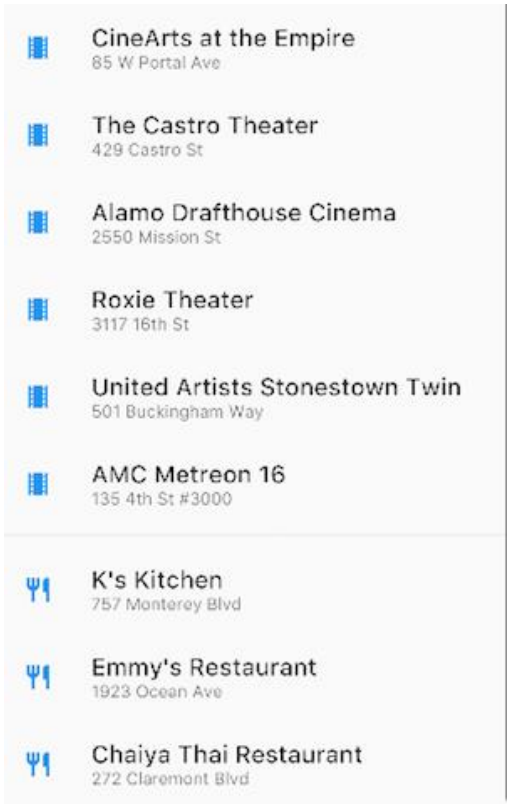
- 2차원 리스트와 같은 레이아웃이 필요할때 사용
- 컨텐츠가 장치의 화면을 넘어설만큼 길다면 자동으로 스코롤 기능 작동
- GridView.count : 컬럼의 수를 지정
- GridView.extent : 타일의 최대 width(pixel) 지정




```
Widget _buildGrid() => GridView.extent(
    maxCrossAxisExtent: 150,
    padding: const EdgeInsets.all(4),
    mainAxisSpacing: 4,
    crossAxisSpacing: 4,
    children: _buildGridTileList(30));

// The images are saved with names pic0.jpg, pic1.jpg...pic29.jpg.
// The List.generate() constructor allows an easy way to create
// a list when objects have a predictable naming pattern.

List<Container> _buildGridTileList(int count) => List.generate(
    count, (i) => Container(child: Image.asset('images/pic$i.jpg')));
```

Flutter basic : Common layout widgets : ListView

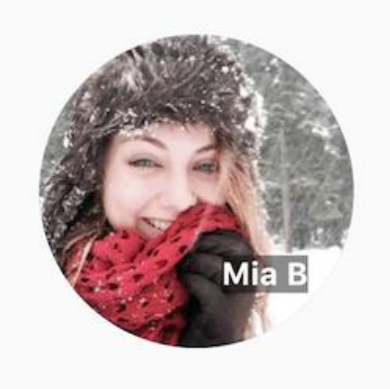
- Column과 비슷한 위젯
- 수평, 수직 리스트 모두 가능
- 리스트가 장치화면보다 커지면 자동으로 스크롤 기능 작동

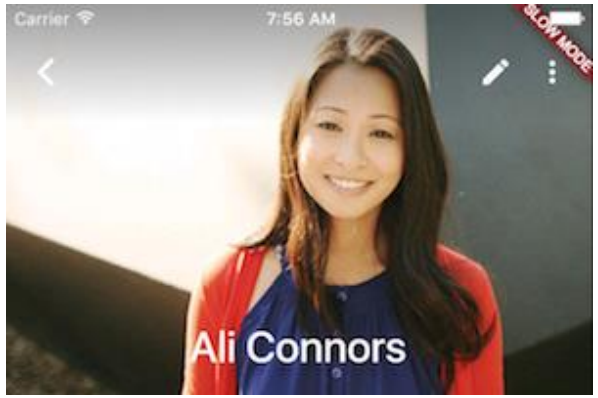



```
Widget _buildList() {
  return ListView(
    children: [
      _tile('CineArts at the Empire', '85 W Portal Ave', Icons.theaters),
      _tile('The Castro Theater', '429 Castro St', Icons.theaters),
      _tile('Alamo Drafthouse Cinema', '2550 Mission St', Icons.theaters),
      _tile('Roxie Theater', '3117 16th St', Icons.theaters),
      _tile('United Artists Stonestown Twin', '501 Buckingham Way',
          Icons.theaters),
      _tile('AMC Metreon 16', '135 4th St #3000', Icons.theaters),
      const Divider(),
      _tile('K₩'s Kitchen', '757 Monterey Blvd', Icons.restaurant),
      _tile('Emmy₩'s Restaurant', '1923 Ocean Ave', Icons.restaurant),
      _tile(
          'Chaiya Thai Restaurant', '272 Claremont Blvd', Icons.restaurant),
      _tile('La Ciccia', '291 30th St', Icons.restaurant),
    ],
  );
}

ListTile _tile(String title, String subtitle, IconData icon) {
  return ListTile(
    title: Text(title,
        style: const TextStyle(
          fontWeight: FontWeight.w500,
          fontSize: 20,
        )),
    subtitle: Text(subtitle),
    leading: Icon(
      icon,
      color: Colors.blue[500],
    ),
  );
}
```

Flutter basic : Common layout widgets : Stack

- 위젯 위에 다른 위젯을 놓고 싶을때 사용 : 이미지 위젯에서 많이 사용됨
- 스택 리스트에서 최초로 놓여지는 위젯이 base 위젯. 나머지 위젯들은 base위젯 위에 놓여진다
- 스택에 놓여진 컨텐츠는 스코롤 할 수 없다.
- Render상자를 넘어서는 이미지는 잘라낼수 있다(선택)
- 예: CircleAvatar위에 배경있는 Text위젯 사용
- 예: 이미지위에 gradient를 놓아 메뉴가 차별화 되도록 처리

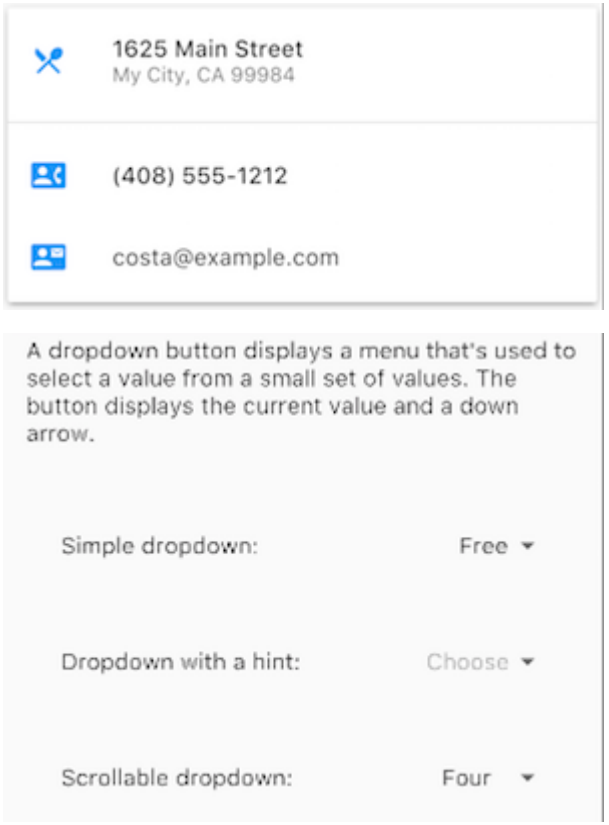





```
Widget _buildStack() {
  return Stack(
    alignment: Alignment(0.6, 0.6),
    children: [
      CircleAvatar(
        backgroundImage: AssetImage('images/pic.jpg'),
        radius: 100,
      ),
      Container(
        decoration: BoxDecoration(
          color: Colors.black45,
        ),
        child: Text(
          'Mia B',
          style: TextStyle(
            fontSize: 20,
            fontWeight: FontWeight.bold,
            color: Colors.white,
          ),
        ),
      ),
    ],
  );
}
```

Flutter basic : Common layout widgets : Card, ListTile

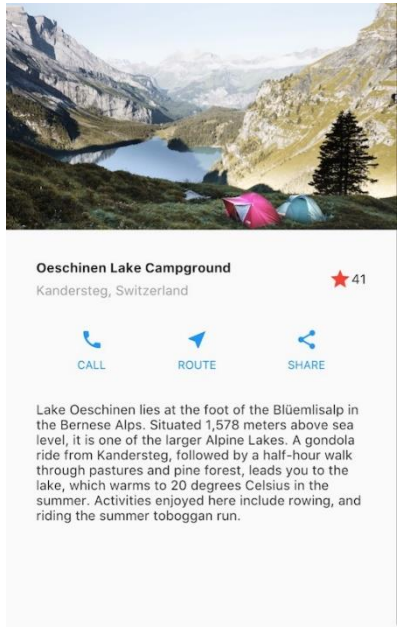
- 정보를 카드 형태로 보여줄 때 사용
- Single child : Row, Column... 등의 children list를 갖는 위젯도 사용가능
- 카드의 내용은 스코롤 할 수 없다.
- SizeBox위젯을 사용하여 카드의 크기를 조절 할 수 있다.
- ListTile : 최대 3줄의 내용과 leading, trailing 아이콘으로 구성



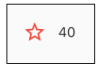

```
Widget _buildCard() {
  return SizedBox(
    height: 210,
    child: Card(
      child: Column(
        children: [
          ListTile(
            title: Text(
              '1625 Main Street',
              style: TextStyle(fontWeight: FontWeight.w500),
            ),
            subtitle: Text('My City, CA 99984'),
            leading: Icon(
              Icons.restaurant_menu,
              color: Colors.blue[500],
            ),
          ),
          Divider(),
          ListTile(
            title: Text(
              '(408) 555-1212',
              style: TextStyle(fontWeight: FontWeight.w500),
            ),
            leading: Icon(
              Icons.contact_phone,
              color: Colors.blue[500],
            ),
          ),
          ListTile(
            title: Text('costa@example.com'),
            leading: Icon(
              Icons.contact_mail,
              color: Colors.blue[500],
            ),
          ),
        ],
      ),
    ),
  );
}
```

Flutter basic : stateless, statefull 위젯 개념

- 위젯은 stateless or stateful 이다.
- 만약 사용자와 상호작용하여 위젯이 변한다면 그것은 stateful 위젯이다.
  - Checkbox, radio, inkwell, …
- Stateless 위젯은 결코 변하지 않는다.
  - Icon, IconButton, Text, …
- Stateful widget
  - 두개의 클래스 StatefulWidget, State를 사용하여 구현
  - 위젯의 state(상태)가 변하면 state object setState()을 호출하고 framework는 화면을 다시 그린다
  - 먼저 State관리를 어디서 할지를 결정



Favorited



Not favorited

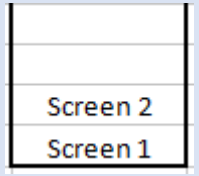
```dart
class FavoriteWidget extends StatefulWidget {
  const FavoriteWidget({Key? key}) : super(key: key);

  @override
  _FavoriteWidgetState createState() => _FavoriteWidgetState();
}

class _FavoriteWidgetState extends State<FavoriteWidget> {
  bool _isFavorited = true;
  int _favoriteCount = 41;

  @override
  Widget build(BuildContext context) {
    return Row(
      mainAxisSize: MainAxisSize.min,
      children: [
        Container(
          padding: EdgeInsets.all(0),
          child: IconButton(
            padding: EdgeInsets.all(0),
            alignment: Alignment.centerRight,
            icon: (_isFavorited ? Icon(Icons.star) : Icon(Icons.star_border)),
            color: Colors.red[500],
            onPressed: _toggleFavorite,
          ),
        ),
        SizedBox(
          width: 18,
          child: SizedBox(
            child: Text('$_favoriteCount'),
          ),
        ),
      ],
    );
  }
}
```

```dart
void _toggleFavorite() {
  setState(() {
    if (_isFavorited) {
      _favoriteCount -= 1;
      _isFavorited = false;
    } else {
      _favoriteCount += 1;
      _isFavorited = true;
    }
  });
}
```

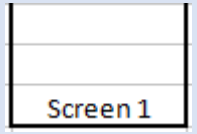
Flutter basic : navigation and routing

- 화면(페이지) 전환을 관리
- 버전별로 차이
  - 1.0
  - 2.0 : Web에서의 페이지 전환을 고려
- Push, pop
  - Push : 스택 맨위에 요소추가
  - Pop : 스택 맨위에서 요소 제거
  - https://github.com/PoojaB26/NavigatorsDemo-Flutter/blob/master/lib/main.dart
- Named route를 사용하는 경우 인수전달은 Arguments 속성을 통해서.
- 전달받은 파라메터의 추출은 두가지 방법이 가능
  - 생성자를 통해서 파라메터 추출
    - 이경우 사전에 MaterialApp의 onGenerateRoute항목에 인수를 받아서 전달하는 로직이 구현되어 있어야 한다.
  - 차이점 : 전달받은 파라메터가 Build 이전에 필요한가?, 이후에 필요한가?
- 페이지간에 데이터 전달

Navigator.of(context).pushNamed('/screen2');



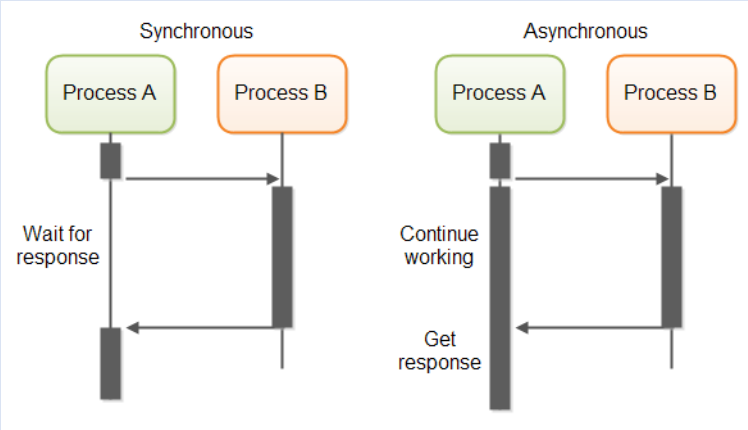
Navigator.of(context).pop();



1. Animate a widget across screens
2. Navigate to a new screen and back
3. Navigate with named routes
4. Pass arguments to a named route
5. Return data from a screen
6. Send data to a new screen

Flutter basic : 비동기

- Flutter 는 단일 thread(프로세스 내에서 실행되는 흐름의 단위)방식
- 비동기 : 외부 데이터를 CRUD하는 동안에도 사용자가 자연스럽게 앱을 사용하도록 돕기위함
  - 네트워크를 통해 데이터를 가져올때
  - DB에 쓰기할때
  - 파일에서 데이터 읽기
- 비동기를 적용하려면 Future class, async, await 키워드 사용
  - 비동기 함수를 만들려면 함수의 body에 async 추가
  - Await 키워드는 async 함수에서만 사용가능
- 동기 동작 : 동작이 완료될 때까지 다른 동작은 멈춤
- 비동기 동작 : 동작이 완료되기 전에도 다른 동작을 허용
- https://dart.dev/codelabs/async-await
- future(소문자 f) : Future(대문자 F)의 인스탄스
  - 비동기 동작 결과를 표현
  - 두 상태중의 하나가 될 수 있다 : uncompleted, completed
  - future를 리턴하는 함수를 호출하면, 함수는 작업 할 것을 큐에 적재하고 uncompleted를 리턴
  - 동작이 완료되면 값을 리턴하던가 에러를 리턴



Flutter basic : 저장장치 : SharedPreference, SQLite

- SharedPreference
  - Key-value 형태의 데이터를 디스크에 저장 : 예) 로그인때 ID/PW등을 저장
  - 단순형태의 데이터 저장에 주로 사용
  - 앱을 종료해도 값이 유지 : 파일에 저장되므로

- SQLite
  - 작은 SQL DB
  - SharedPreference에 비해 좀더 복잡한 데이터 저장에 사용
  - Select * from name_table where id='korea' 등과같은 기본 sql 쿼리사용가능

1. Store key-value data on disk
2. Persist data with SQLite
3. Read and write files

Flutter basic : Networking & http

```
<manifest xmlns:android...>
...
<uses-permission android:name="android.permission.INTERNET" /> <application ...
</manifest>
```

- 네트워크를 사용하려면 androidMainfest.xml파일에 퍼미션적용이 필요
- Backend : https://jsonplaceholder.typicode.com
  - 예) 기상청 API

- Delete data on the internet
- Fetch data from the internet
- Make authenticated requests
- Parse JSON in the background
- Send data to the internet
- Update data over the internet

Flutter basic : JSON & serialization

- encoding : 데이터구조를 문자열로 바꾸는 것 (서버에서 모바일로 데이터 전송때)
- Decoding : encoding의 반대, 문자열을 데이터 구조로 변경
- Serialization :데이터 구조를 보다 읽기쉬운 형식으로 변환하는과정
  - 메뉴얼방식 vs 코드자동생성방식
  - 소규모 프로젝트 -> 수동, 중대형 프로젝트 -> 자동
- JSON(Java Script Object Notation) : 데이터를 저장, 전송할때 많이 사용하는 경량 DATA 교환 형식 – 데이터 표형 방식의 하나
- 사람, 컴퓨터 모두 이해하고 쉽고 용량이 작아서 많이 사용
- Key-value형태도 가능
- 문자열은 항상 따옴표를 사용
- JSON 변환 도구 : https://app.quicktype.io/

```
{
  "firstName": "Kwon",
  "lastName": "YoungJae",
  "email": "kyoje11@gmail.com",
  "hobby": ["puzzles","swimming"],
  “favorite numbers”, [7, 12, 100],
  “single” : true
}
```

https://docs.flutter.dev/development/data-and-backend/json#code-generation

1. Serializing JSON inline

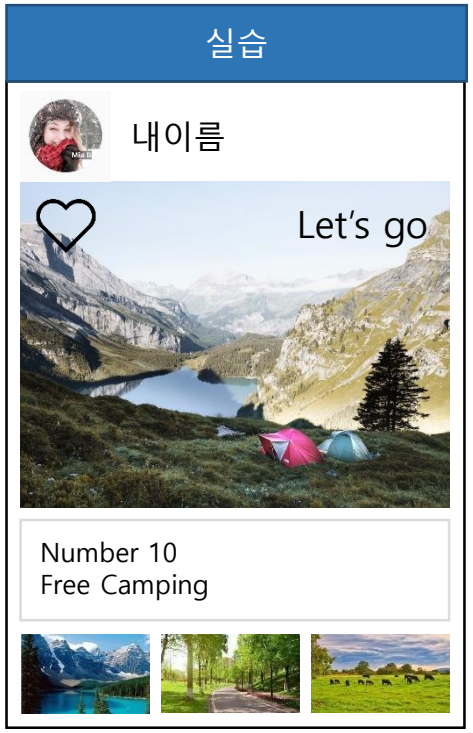
```
Map<String, dynamic> user = jsonDecode(jsonString); // 런타임전에는 타입을 알수없다
print('Howdy, ${user['name']}!');
print('We sent the verification link to ${user['email']}.');
```

2. Serializing JSON inside model classes

```
class User {
  final String name;
  final String email;

  User(this.name, this.email);

  User.fromJson(Map<String, dynamic> json)
      : name = json['name'],
        email = json['email'];

  Map<String, dynamic> toJson() => {
        'name': name,
        'email': email,
      };
}

Map<String, dynamic> userMap = jsonDecode(jsonString);
var user = User.fromJson(userMap);

print('Howdy, ${user.name}!');
print('We sent the verification link to ${user.email}.');
```

실습

- **Codelabs**
  - Good for beginners
    - Basic Flutter layout concepts
    - Write your first Flutter app, part 1
    - Write your first Flutter app, part 2
    - Building beautiful UIs with Flutter
    - 과제 : UI, setState, 화면전환
  - Designing a Flutter UI
    - MDC-101 Flutter: Material Components (MDC) Basics
    - MDC-102 Flutter: Material Structure and Layout
    - MDC-103 Flutter: Material Theming with Color, Shape, Elevation, and Type
    - MDC-104 Flutter: Material Advanced Components



1. 화면 레이아웃
2. 하트 클릭하면 색깔 바뀌게