twitch.tv/dancojocar
youtube.com/dancojocar

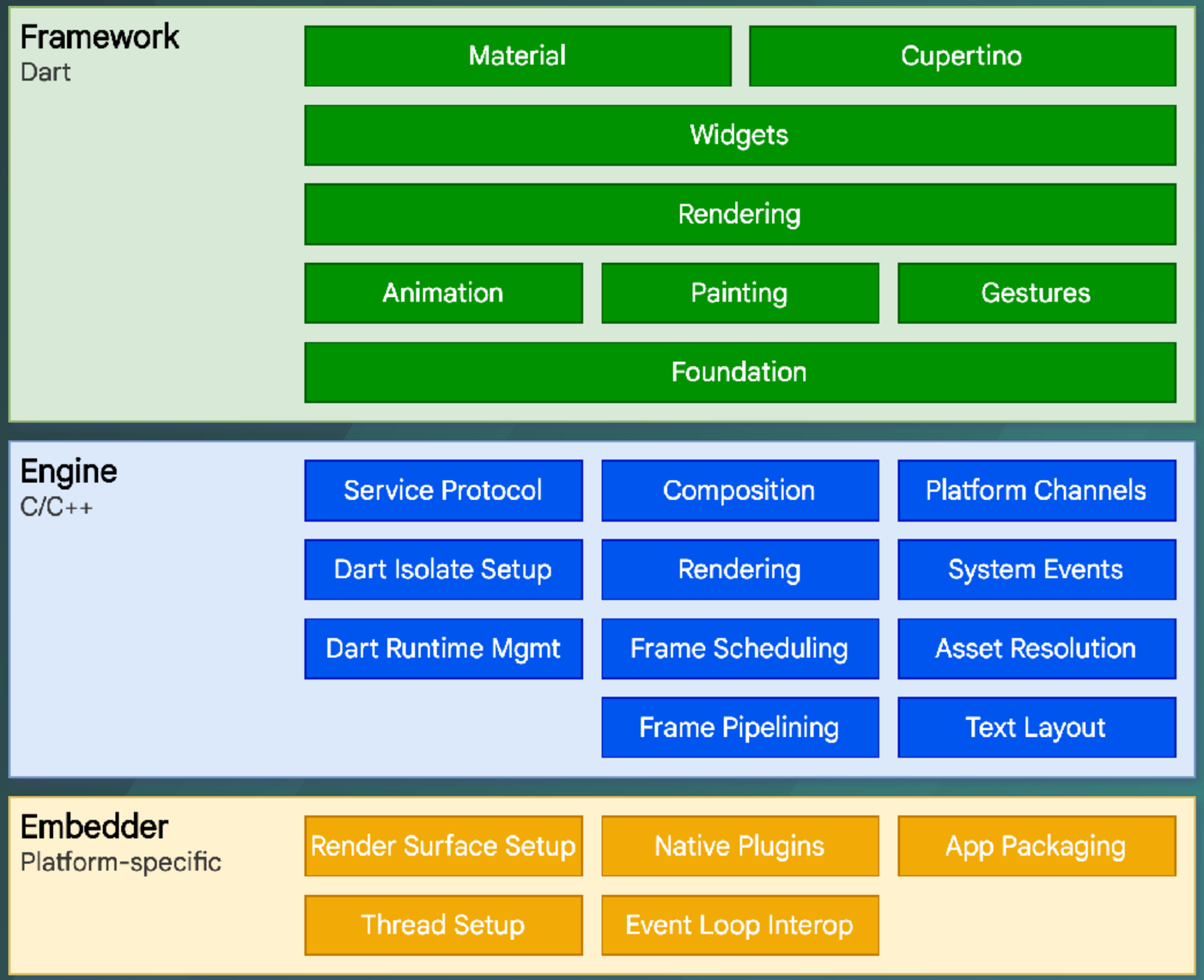
Lecture #6
Flutter Architecture

Mobile Applications
Fall 2024

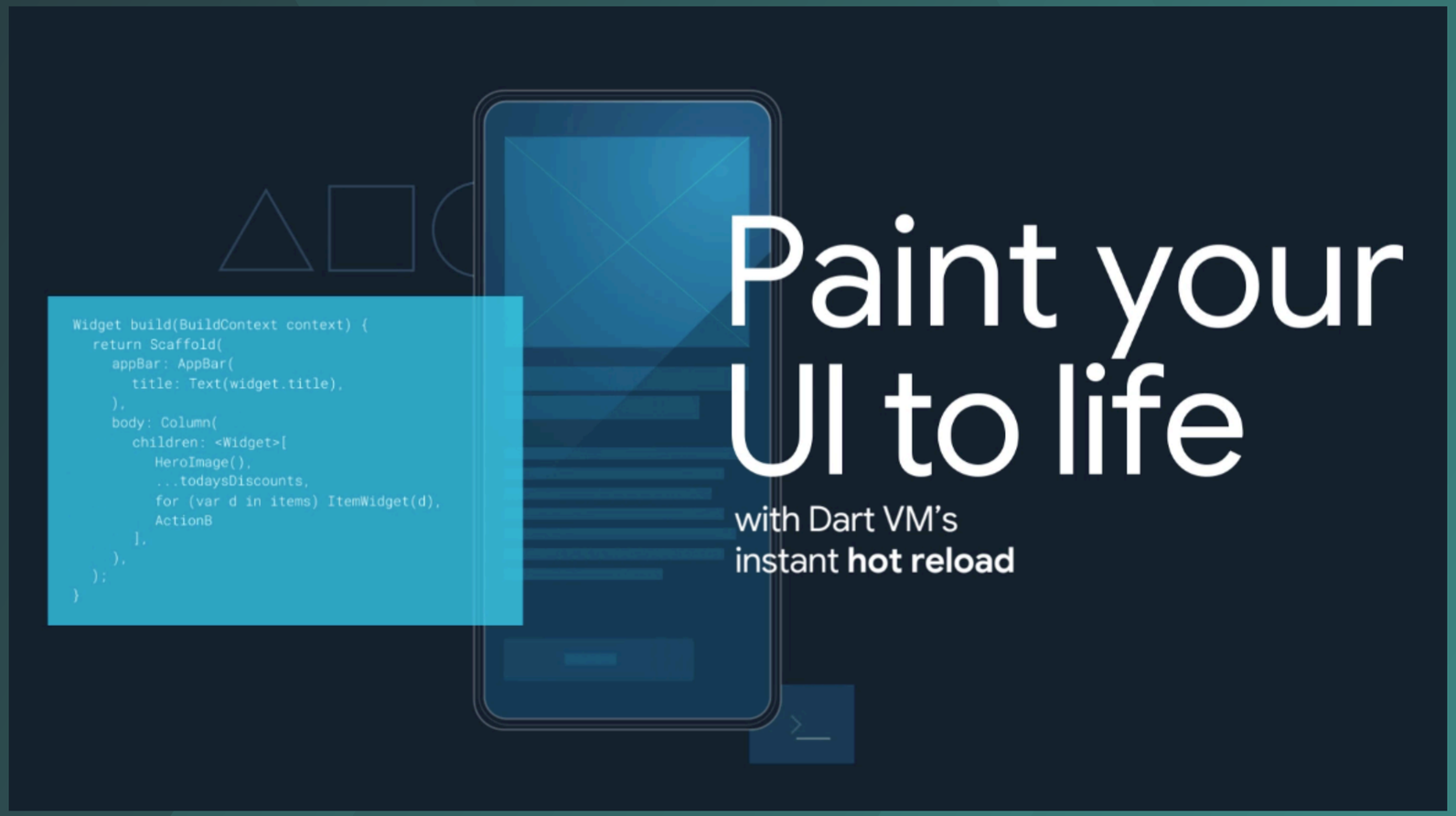
Flutter - Architecture Application

- Widgets
- Gestures
- Concept of State
- Layers

Architectural Layers



Dart


Widget build(BuildContext context) {
return Scaffold(
appBar: AppBar(
title: Text(widget.title),
),
body: Column(
children: <Widget>[
HeroImage(),
...todaysDiscounts,
for (var d in items) ItemWidget(d),
ActionB
],
),
);
}
Paint your UI to life
with Dart VM's instant hot reload

Dart

Optimized for UI

Develop with a programming language specialized around the needs of user interface creation

Productive development

Make changes iteratively: use hot reload to see the result instantly in your running app

Fast on all platforms

Compile to ARM & x64 machine code for mobile, desktop, and backend. Or compile to JavaScript for the web

Optimized
for UI

Dart

Optimized for UI

- Mature and complete **async-await** for user interfaces containing event-driven code, paired with **isolate-based concurrency**
- A programming language optimized for building user interfaces with features such as **sound null safety**, the **spread operator** for expanding collections, and **collection if** for customizing UI for each platform
- A programming language that is easy to learn, with a **familiar syntax**

35°

Mostly Sunny

```
fetchTemperature() async {
  // This call is non-blocking
  // The UI thread will continue
  // to render with no locks
  final response = await
    http.get('https://my/weather');

  if (response.statusCode == 200) {
    temp.setText(response.body);
  } else {
    temp.setText('Unknown temp.');
  }
}
```

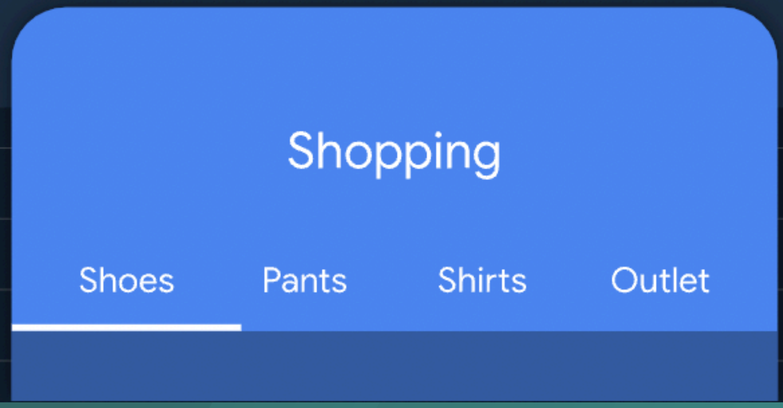
Optimized
for UI

Dart

Optimized for UI

- Mature and complete **async-await** for user interfaces containing event-driven code, paired with **isolate-based concurrency**
- A programming language optimized for building user interfaces with features such as **sound null safety**, the **spread operator** for expanding collections, and **collection if** for customizing UI for each platform
- A programming language that is easy to learn, with a **familiar syntax**

```
TabBar build(BuildContext context) {
  return TabBar(tabs: [
    Tab(text: 'Shoes'),
    Tab(text: 'Pants'),
    Tab(text: 'Shirts'),
    if (promoActive) Tab(text: 'Outlet'),
  ]);
}
```



Optimized
for UI

Dart

Optimized for UI

- Mature and complete **async-await** for user interfaces containing event-driven code, paired with **isolate-based concurrency**
- A programming language optimized for building user interfaces with features such as **sound null safety**, the **spread operator** for expanding collections, and **collection if** for customizing UI for each platform

A programming language that is easy to learn, with a **familiar syntax**

Dart

```
class Segment {
  int links = 4;
  toString() => "I have $links links";
}
```

Kotlin

```
class Segment {
  var links: Int = 4
  override fun toString()= "I have $links links"
}
```

Swift

```
class Segment: CustomStringConvertible  {
  var links: Int = 4
  public var description: String { return
  "I have \(links) links"}
}
```

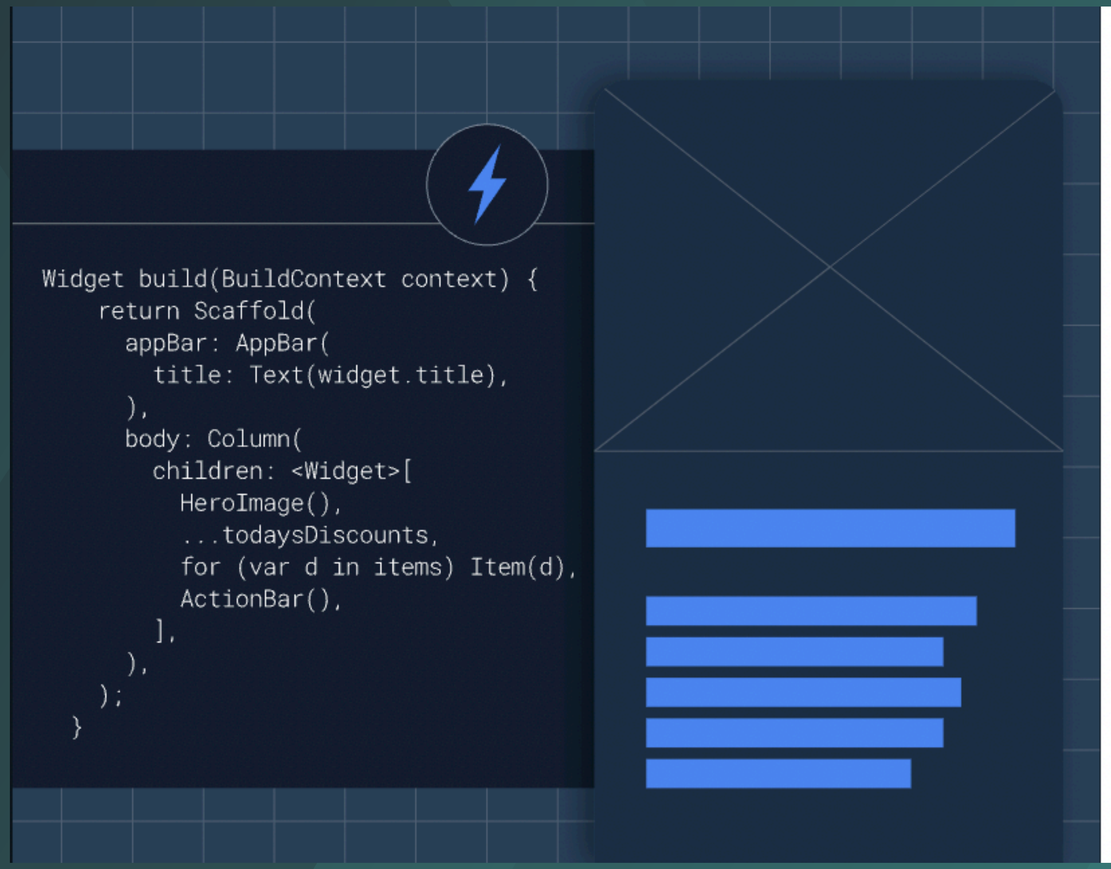
TS

TypeScript

```
class Segment {
  links: number = 4
  public toString = () : string => { return
  `I have ${this.links} links` };
}
```

Productive development

Dart



Productive development

- Make changes to your source code iteratively, using hot reload to instantly see the effect in the running app
- Write code using a flexible type system with rich static analysis and powerful, configurable tooling
- Do profiling, logging, and debugging with your code editor of choice

Productive development

Dart

```
//The type of temperature is inferred to be int.
var temperature = 25;

// Static code analysis catches errors early.
temperature = 'Freezing';
```

A String can't be assigned to an 'int'

```
// Customizable code style checks
class weather{}
```

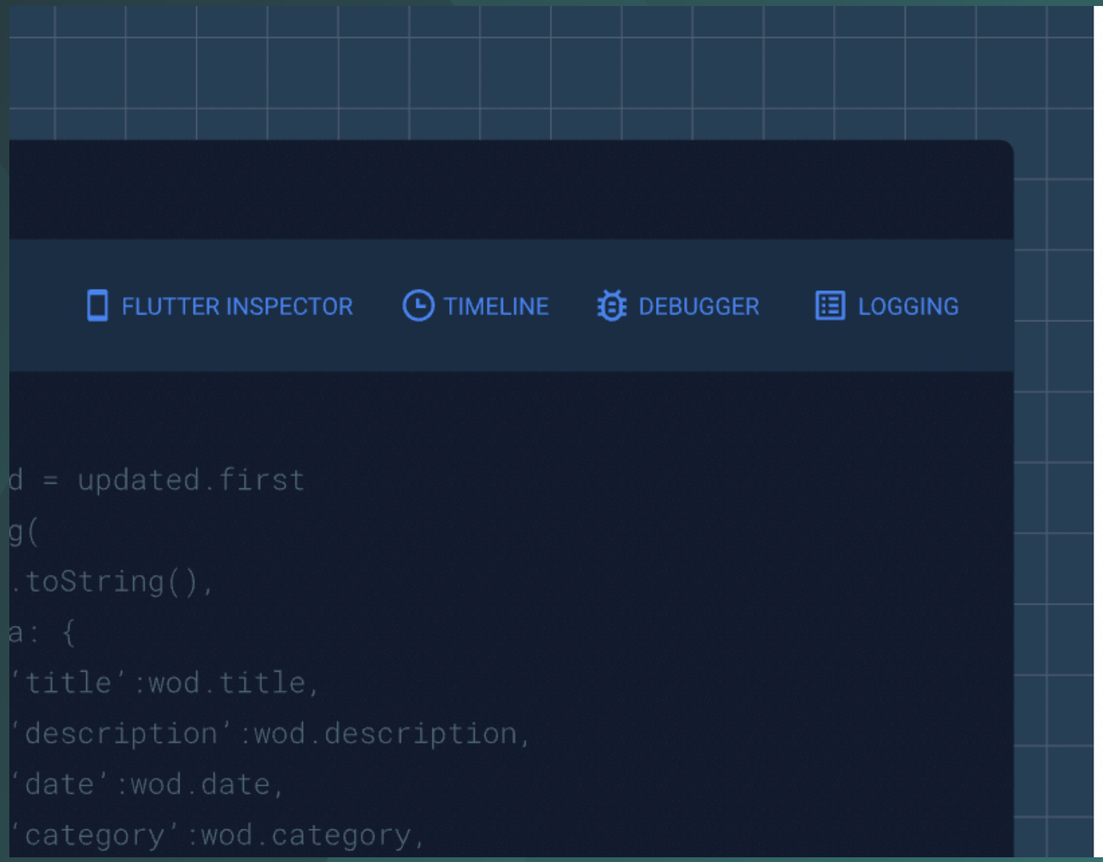
[dart] Name types using UpperCamelCase

Productive development

- Make changes to your source code iteratively, using **hot reload** to instantly see the effect in the running app
- Write code using a flexible type system with rich static analysis and powerful, **configurable tooling**
- Do **profiling**, **logging**, and **debugging** with your code editor of choice

Productive development

Dart



Productive development

- Make changes to your source code iteratively, using hot reload to instantly see the effect in the running app
- Write code using a flexible type system with rich static analysis and powerful, configurable tooling

Do profiling, logging, and debugging with your code editor of choice

Fast on all platforms

Dart

Fast on all platforms

- AOT-compile apps to native machine code for instant startup
- Target the web with complete, mature, fast compilers for JavaScript
- Run backend code supporting your app, written using a single programming language

```
$ dart compile exe hello.dart
$ time ./hello.exe
Hello, Developer!

Real 0m0.012s
# Instant startup; completed in just 12ms
$
```

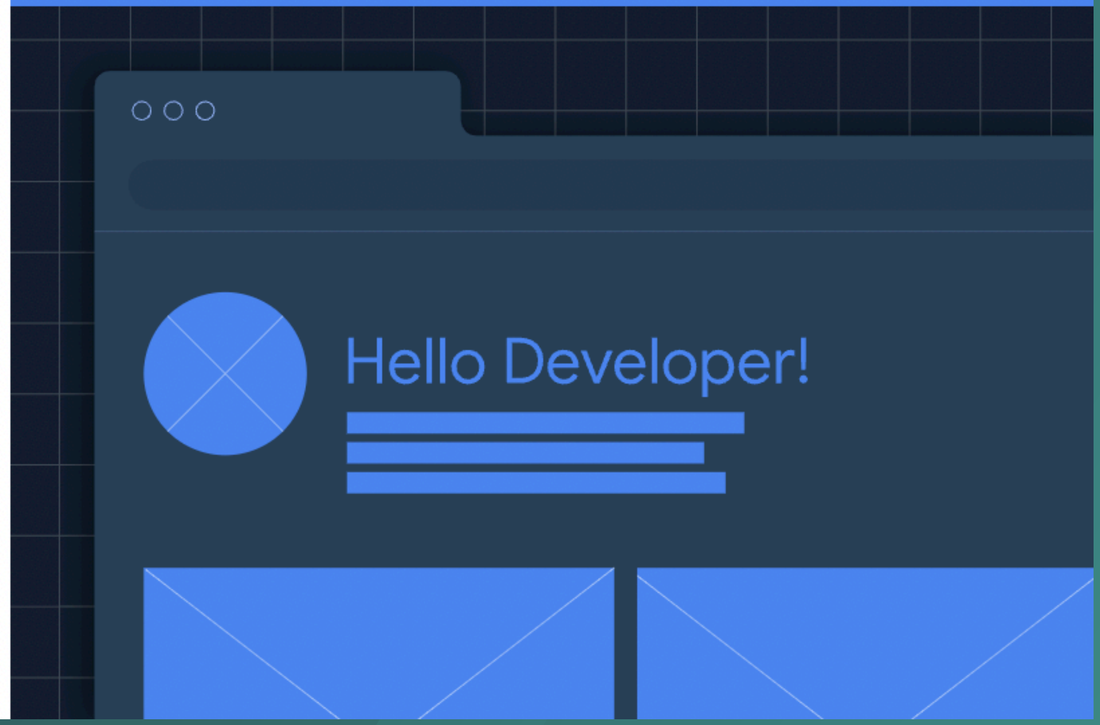
Fast on all platforms

Dart

Fast on all platforms

- AOT-compile apps to native machine code for instant startup
- Target the web with complete, mature, fast compilers for JavaScript
- Run backend code supporting your app, written using a single programming language

```
$ webdev serve      # Serve for development
$ webdev build      # Production build
```



Fast on all platforms

Dart

Fast on all platforms

- AOT-compile apps to native machine code for instant startup
- Target the web with complete, mature, fast compilers for JavaScript
- Run backend code supporting your app, written using a single programming language

AOT-compile for short-lived “serverless” jobs where startup time is critical

Variables and Data Types

```dart
var name = 'Dart';
```

Variables and Data Types

```
void main() {
  final a = 12;
  const pi = 3.14;
  print(a);
  print(pi);
}
```

Variables and Data Types

- Numbers
- Strings
- Booleans
- Lists and Maps

```
void main() {
  var list = [1,2,3,4,5];
  print(list);
}
```

Variables and Data Types

- Numbers
- Strings
- Booleans
- Lists and Maps

```
void main() {
  var mapping = {'id': 1,'name':'Dart'};
  print(mapping);
}
```

Variables and Data Types

- Numbers
- Strings
- Booleans
- Lists and Maps
- Dynamic

```dart
void main() {
  dynamic name = "Dart";
  print(name);
}
```

Decision Making and Loops

```
void main(){
    //read number from user
    print('Enter a number');
    var a = double.parse(stdin.readLineSync()!);

    if(a<0){
        print('$a is negative number.');
    } else if(a==0) {
        print('$a is zero. Neither negative nor positive');
    } else {
        print('$a is positive number.');
    }
}
```

Decision Making and Loops

```
1 + 1 == 2 ? print('check true') : print('check false');
//same with
if ( 1 + 1 == 2) {
    print('check true');
else {
    print('check false');
}
```

Decision Making and Loops

```
void main() {
  int n = 3;
  switch (n) {
    case 1:
      print("Value is 1");
      break;
    case 2:
      print("Value is 2");
      break;
    default:
      print("Out of range");
      break;
    }
}
```

Decision Making and Loops

```
void main() {
  var i = 0;
  while (i < 5) {
    print('Hello World');
    i++;
  }
}
```

Decision Making and Loops

```
void main() {
  var i = 0;
  do {
    print('Hello World');
    i++;
  } while (i < 5);
}
```

Decision Making and Loops

```
void main(){
   var n = 6;
   var factorial = 1;

   //for loop to calculate factorial
   for(var i=2; i<=n; i++) {
      factorial = factorial*i;
   }

   print('Factorial of ${n} is ${factorial}');
}
```

Decision Making and Loops

```
void main() {
  for (var i = 1; i < 8; i++) {
    if (i == 5) {
      break;
    }
    print(i);
  }
}
```

Decision Making and Loops

```
void main() {
  for (var i = 1; i < 8; i++) {
    if (i == 5) {
      break;
    }
    print(i);
  }
}
```

```
void main() {
  var i = 0;
  while (i < 7) {
    i++;
    if (i == 5) {
      break;
    }
    print(i);
  }
}
```

```
void main() {
  var i = 0;
  do {
    i++;
    if (i == 5) {
      break;
    }
    print(i);
  } while (i < 7);
}
```

Decision Making and Loops

```
void main() {
  for (var i = 1; i < 8; i++) {
    if (i == 5) {
      continue;
    }
    print(i);
  }
}
```

```
void main() {
  var i = 0;
  while (i < 7) {
    i++;
    if (i == 5) {
      continue;
    }
    print(i);
  }
}
```

```
void main() {
  var i = 0;
  do {
    i++;
    if (i == 5) {
      continue;
    }
    print(i);
  } while (i < 7);
}
```

Null Operator

```
void main() {
  print(0 ?? 1);  // <- 0
  print(1 ?? null);  // <- 1
  print(null ?? null); // <- null
  print(null ?? null ?? 2); // <- 2
}
```

Null-aware Assignment

```
void main() {
  int value;
  print(value); // <- null
  value ??= 5;
  print(value); // <- 5, changed from null
  value ??= 6;
  print(value); // <- 5, no change
}
```

Null-aware Access

```
void main() {
  String value; // <- value is null
  print(value.toLowerCase()); // <- will crash
  print(value?.toLowerCase().toUpperCase());  // <- will crash
  print(value?.toLowerCase()?.toUpperCase()); // <- output is null
}
```

Null-aware Spread Operator

```
void main() {
  List<int> list = [1, 2, 3];
  List<String> list2; // <- list2 is null
  print(['chocolate', ...?list2]); // <- [chocolate]
  print([0, ...?list2, ...list]); // <- [0, 1, 2, 3]
  print(['cake!', ...list2]);  // <- will crash
}
```

Functions

```
void main() {
  add(3, 4);
}
void add(int a, int b) {
  int c;
  c = a + b;
  print(c);
}
```

OOP

```
class Employee {
  String name;

  //getter method
  String get empName {
    return name;
  }
  //setter method
  void set empName(String name) {
    this.name = name;
  }
  //function definition
  void result() {
    print(name);
  }
}
```

```
void main() {
  //object creation
  Employee emp = new Employee();
  emp.name = "employee1";
  emp.result(); //function call
}
```

OOP

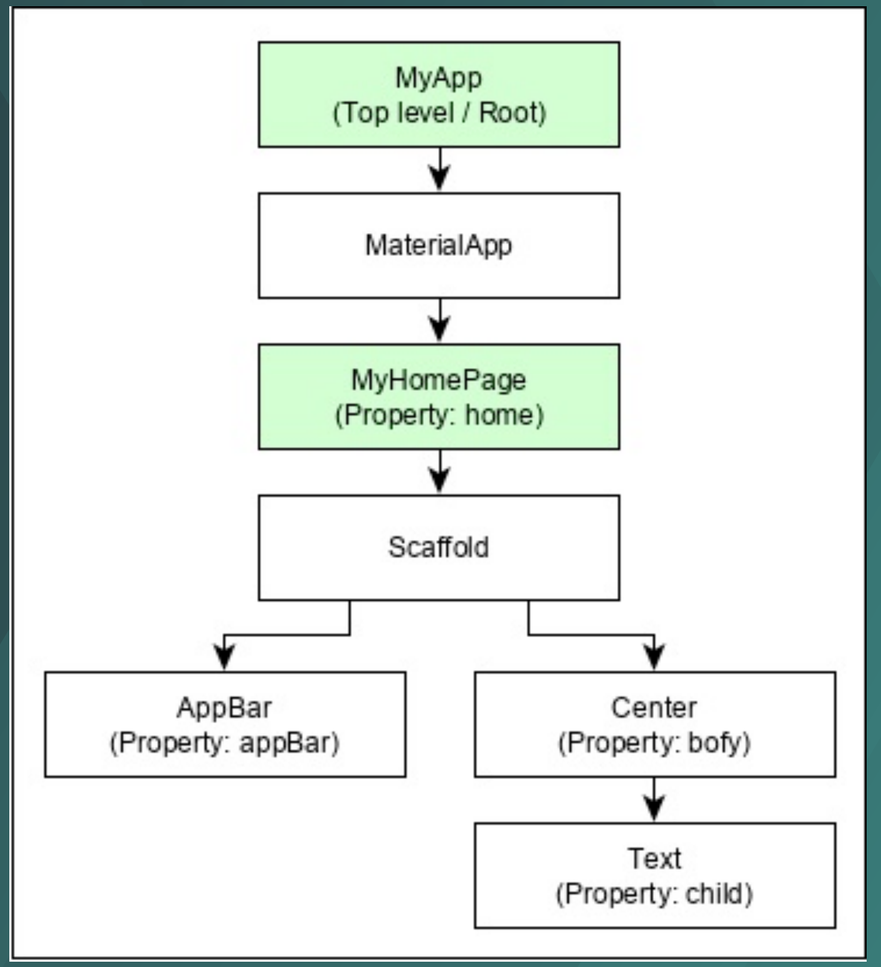
```
class Employee {
  String name;

  //getter method
  String get empName {
    return name;
  }
  //setter method
  void set empName(String name) {
    this.name = name;
  }
  //function definition
  void result() {
    print(name);
  }
}
```

```
void main() {
  //object creation
  Employee emp = Employee();
  emp.name = "employee1";
  emp.result(); //function call
}
```

Widgets

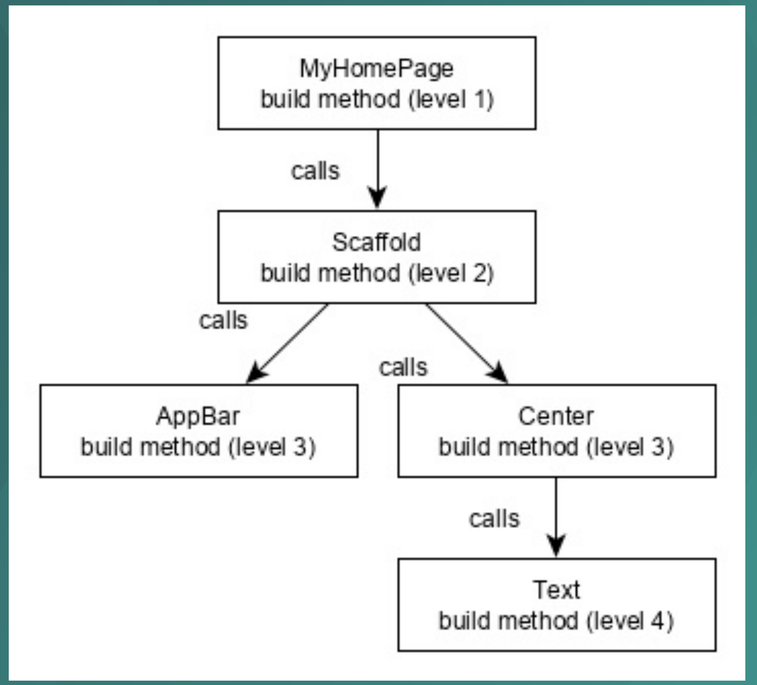
- StatelessWidget
- StatefullWidget

Widgets


MyApp
(Top level / Root)
MaterialApp
MyHomePage
(Property: home)
Scaffold
AppBar
(Property: appBar)
Center
(Property: bofy)
Text
(Property: child)

Widgets

```
class MyHomePage extends StatelessWidget {
  MyHomePage({Key key, this.title}) : super(key: key);

  final String title;
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Text(this.title), ),
      body: Center(child: Text( 'Hello World',)),
    );
  }
}
```



Text

```
Text('Hello World!', style: TextStyle(fontWeight: FontWeight.bold))
```

Text

```
Text.rich(
  TextSpan(
    children: <TextSpan>[
      TextSpan(text: "Hello ", style:
      TextStyle(fontStyle: FontStyle.italic)),
      TextSpan(text: "World", style:
      TextStyle(fontWeight: FontWeight.bold)),
    ],
  ),
)
```

Image

```
// pubspec.yaml

flutter:
  assets:
    - assets/smiley.png
```

Image

```
// pubspec.yaml

flutter:
  assets:
    - assets/smiley.png
```

```
Image.asset("assets/smiley.png")
```

Image

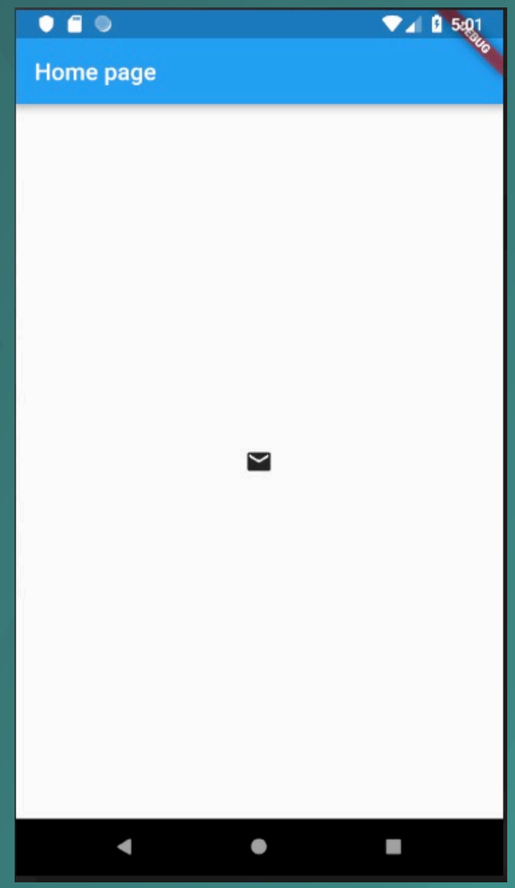
```
class MyHomePage extends StatelessWidget {
  MyHomePage({Key key, this.title}) : super(key: key);
  final String title;

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar( title: Text(this.title), ),
      body: Center( child: Image.asset("assets/smiley.png")),
    );
  }
}
```

Image

```
class MyHomePage extends StatelessWidget {
  MyHomePage({Key key, this.title}) : super(key: key);
  final String title;

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar( title: Text(this.title), ),
      body: Center( child: Image.asset("assets/smiley.png")),
    );
  }
}
```

Icon

```
class MyHomePage extends StatelessWidget {
  MyHomePage({Key key, this.title}) : super(key: key);
  final String title;

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Text(this.title),),
      body: Center( child: Icon(Icons.email)),
    );
  }
}
```

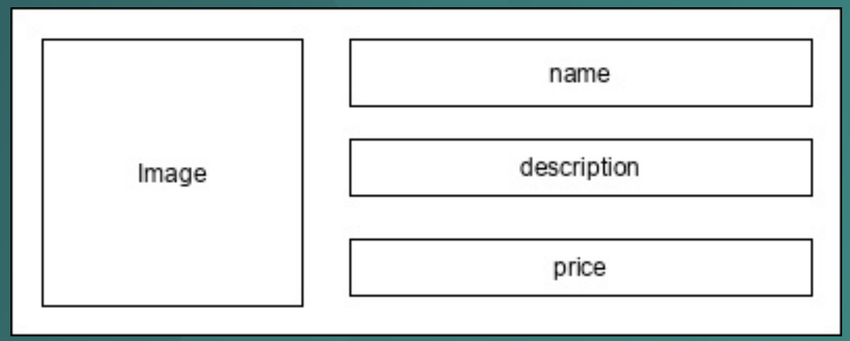


Layout

```
class MyButton extends StatelessWidget {
  MyButton({Key key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Container(
      decoration: const BoxDecoration(
        border: Border(
          top: BorderSide(width: 1.0, color: Color(0xFFFFFFFFFF)),
          left: BorderSide(width: 1.0, color: Color(0xFFFFFFFFFF)),
          right: BorderSide(width: 1.0, color: Color(0xFFFF000000)),
          bottom: BorderSide(width: 1.0, color: Color(0xFFFF000000)),
        ),
      ),
      child: Container(
        padding: const
        EdgeInsets.symmetric(horizontal: 20.0, vertical: 2.0),
        decoration: const BoxDecoration(
          border: Border(
            top: BorderSide(width: 1.0, color: Color(0xFFFFDFDFDF)),
            left: BorderSide(width: 1.0, color: Color(0xFFFFDFDFDF)),
            right: BorderSide(width: 1.0, color: Color(0xFFFF7F7F7F)),
            bottom: BorderSide(width: 1.0, color: Color(0xFFFF7F7F7F)),
          ),
          color: Colors.grey,
        ),
        child: const Text(
          'OK',textAlign: TextAlign.center, style: TextStyle(color: Colors.black)
        ),
      ),
    );
  }
}
```

Layout

```
class ProductBox extends StatelessWidget {
  ProductBox({Key key, this.name, this.description, this.price, this.image}) : super(key: key);
  final String name;
  final String description;
  final int price;
  final String image;
  Widget build(BuildContext context) {
    return Container(
      padding: EdgeInsets.all(2),
      height: 120,
      child: Card(
        child: Row(
          mainAxisAlignment: MainAxisAlignment.spaceEvenly,
          children: <Widget>[
            Image.asset("assets/appimages/" +image),
            Expanded(
              child: Container(
                padding: EdgeInsets.all(5),
                child: Column(
                  mainAxisAlignment: MainAxisAlignment.spaceEvenly,
                  children: <Widget>[
                    Text(this.name, style: TextStyle(fontWeight: FontWeight.bold)),
                    Text(this.description),
                    Text("Price: " + this.price.toString()),
                    ],
                )
              )
            )]
          )
        )
      );
    }
}
```



Gestures

- Tap
  - onTapDown
  - onTapUp
  - onTap
  - onTapCancel
- Double tap
  - onDoubleTap
- Long press
  - onLongPress

Gestures

- Tap
  - onTapDown
  - onTapUp
  - onTap
  - onTapCancel
- Double tap
  - onDoubleTap
- Long press
  - onLongPress
- Vertical drag
  - onVerticalDragStart
  - onVerticalDragUpdate
  - onVerticalDragEnd
- Horizontal drag
  - onHorizontalDragStart
  - onHorizontalDragUpdate
  - onHorizontalDragEnd

Gestures

- Tap
  - onTapDown
  - onTapUp
  - onTap
  - onTapCancel
- Double tap
  - onDoubleTap
- Long press
  - onLongPress
- Vertical drag
  - onVerticalDragStart
  - onVerticalDragUpdate
  - onVerticalDragEnd
- Horizontal drag
  - onHorizontalDragStart
  - onHorizontalDragUpdate
  - onHorizontalDragEnd
- Pan
  - onPanStart
  - onPanUpdate
  - onPanEnd

Gestures

```
body: Center(
  child: GestureDetector(
    onTap: () {
      _showDialog(context);
    },
    child: Text( 'Hello World', )
  )
)
```

Gestures

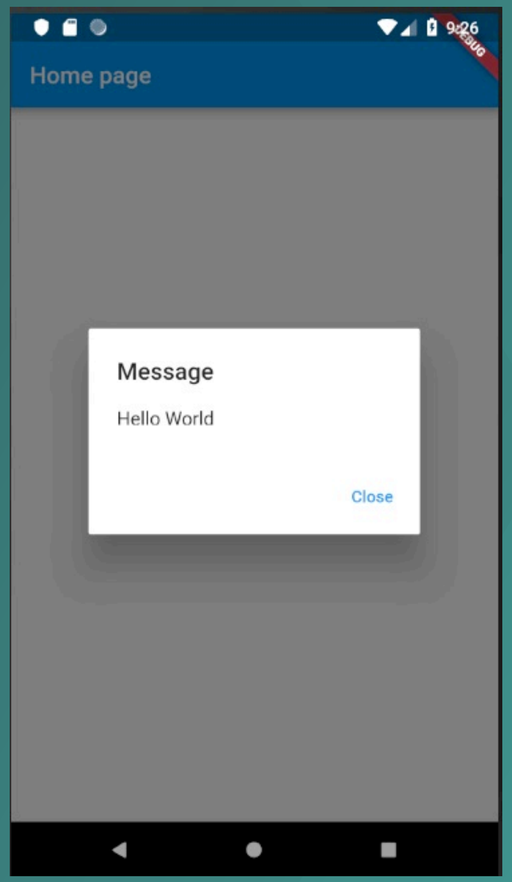
```
body: Center(
  child: GestureDetector(
    onTap: () {
      _showDialog(context);
    },
    child: Text( 'Hello World', )
  )
)
```

```
showDialog(
  context: context, builder: (BuildContext context) {
    // return object of type Dialog
    return AlertDialog(
      title: new Text("Message"),
      content: new Text("Hello World"),
      actions: <Widget>[
        new FlatButton(
          child: new Text("Close"),
          onPressed: () {
            Navigator.of(context).pop();
          },
        ),
      ],
    );
  },
);
```

Gestures

```
body: Center(
  child: GestureDetector(
    onTap: () {
      _showDialog(context);
    },
    child: Text( 'Hello World', )
  )
)
```

```
showDialog(
  context: context, builder: (BuildContext context) {
    // return object of type Dialog
    return AlertDialog(
      title: new Text("Message"),
      content: new Text("Hello World"),
      actions: <Widget>[
        new FlatButton(
          child: new Text("Close"),
          onPressed: () {
            Navigator.of(context).pop();
          },
        ),
      ],
    );
  },
);
```



State Management

- Ephemeral
- Application State

Ephemeral State Management

```
class RatingBox extends StatefulWidget { }

class _RatingBoxState extends State<RatingBox> { }
```

Ephemeral State Management

```
class _RatingBoxState extends State<RatingBox> { }

class RatingBox extends StatefulWidget {
  @override
  _RatingBoxState createState() => _RatingBoxState();
}
```

Ephemeral State Management

```
class _RatingBoxState extends State<RatingBox> { }

class RatingBox extends StatefulWidget {
  @override
  _RatingBoxState createState() => _RatingBoxState();
}
```

```
class _RatingBoxState extends State<RatingBox> {
  int _rating = 0;
  void _setRatingAsOne() {
    setState( () {
      _rating = 1;
    });
  }
  void _setRatingAsTwo() {
    setState( () {
      _rating = 2;
    });
  }
  void _setRatingAsThree() {
    setState( () {
      _rating = 3;
    });
  }
  Widget build(BuildContext context) {
    double _size = 20;
    print(_rating);
    return Row(
      mainAxisAlignment: MainAxisAlignment.end,
      crossAxisAlignment: CrossAxisAlignment.end,
      mainAxisSize: MainAxisSize.max,
      children: <Widget>[
        Container(
          padding: EdgeInsets.all(0),
          child: IconButton(
            icon: (_rating >= 1 ? Icon(Icons.star, size: _size,) :
            Icon(Icons.star_border, size: _size,))
```

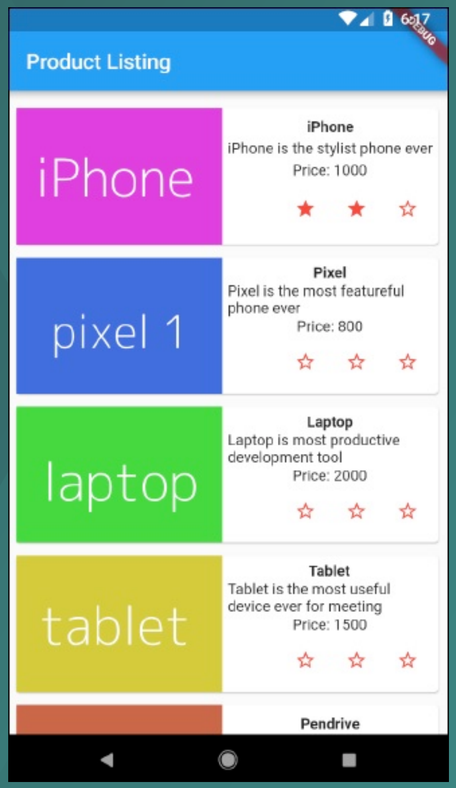
```dart
class RatingBox extends StatefulWidget { }

class _RatingBoxState extends State<RatingBox> { }

class RatingBox extends StatefulWidget {
  @override
  _RatingBoxState createState() => _RatingBoxState();
}
```

```dart
class _RatingBoxState extends State<RatingBox> {
  int _rating = 0;
  void _setRatingAsOne() {
    setState( () {
      _rating = 1;
    });
  }
  void _setRatingAsTwo() {
    setState( () {
      _rating = 2;
    });
  }
  void _setRatingAsThree() {
    setState( () {
      _rating = 3;
    });
  }
  Widget build(BuildContext context) {
    double _size = 20;
    print(_rating);
    return Row(
      mainAxisAlignment: MainAxisAlignment.end,
      crossAxisAlignment: CrossAxisAlignment.end,
      mainAxisSize: MainAxisSize.max,
      children: <Widget>[
        Container(
          padding: EdgeInsets.all(0),
          child: IconButton(
            icon: (_rating >= 1 ? Icon(Icons.star, size: _size,) :
            Icon(Icons.star_border, size: _size,)),
            color: Colors.red[500],
            onPressed: _setRatingAsOne,
            iconSize: _size,
          ),
        ),
        Container(
          padding: EdgeInsets.all(0),
          child: IconButton(
            icon: (_rating >= 2 ? Icon(Icons.star, size: _size,) :
            Icon(Icons.star_border, size: _size,)),
            color: Colors.red[500],
            onPressed: _setRatingAsTwo,
            iconSize: _size,
```

```dart
class RatingBox extends StatefulWidget { }

class _RatingBoxState extends State<RatingBox> { }

class RatingBox extends StatefulWidget {
  @override
  _RatingBoxState createState() => _RatingBoxState();
}
```

```dart
  });
}
Widget build(BuildContext context) {
  double _size = 20;
  print(_rating);
  return Row(
    mainAxisAlignment: MainAxisAlignment.end,
    crossAxisAlignment: CrossAxisAlignment.end,
    mainAxisSize: MainAxisSize.max,
    children: <Widget>[
      Container(
        padding: EdgeInsets.all(0),
        child: IconButton(
          icon: (_rating >= 1 ? Icon(Icons.star, size: _size,) :
          Icon(Icons.star_border, size: _size,)),
          color: Colors.red[500],
          onPressed: _setRatingAsOne,
          iconSize: _size,
        ),
      ),
      Container(
        padding: EdgeInsets.all(0),
        child: IconButton(
          icon: (_rating >= 2 ? Icon(Icons.star, size: _size,) :
          Icon(Icons.star_border, size: _size,)),
          color: Colors.red[500],
          onPressed: _setRatingAsTwo,
          iconSize: _size,
        ),
      ),
      Container(
        padding: EdgeInsets.all(0),
        child: IconButton(
          icon: (_rating >= 3 ? Icon(Icons.star, size: _size,) :
          Icon(Icons.star_border, size: _size,)),
          color: Colors.red[500],
          onPressed: _setRatingAsThree,
          iconSize: _size,
        ),
      ),
    ],
  );
}
}
```

```
    setState(() {
      _rating = 3;
    });
  }
  Widget build(BuildContext context) {
    double _size = 20;
    print(_rating);
    return Row(
      mainAxisAlignment: MainAxisAlignment.end,
      crossAxisAlignment: CrossAxisAlignment.end,
      mainAxisSize: MainAxisSize.max,
      children: <Widget>[
        Container(
          padding: EdgeInsets.all(0),
          child: IconButton(
            icon: (_rating >= 1 ? Icon(Icons.star, size: _size,) :
            Icon(Icons.star_border, size: _size,)),
            color: Colors.red[500],
            onPressed: _setRatingAsOne,
            iconSize: _size,
          ),
        ),
        Container(
          padding: EdgeInsets.all(0),
          child: IconButton(
            icon: (_rating >= 2 ? Icon(Icons.star, size: _size,) :
            Icon(Icons.star_border, size: _size,)),
            color: Colors.red[500],
            onPressed: _setRatingAsTwo,
            iconSize: _size,
          ),
        ),
        Container(
          padding: EdgeInsets.all(0),
          child: IconButton(
            icon: (_rating >= 3 ? Icon(Icons.star, size: _size,) :
            Icon(Icons.star_border, size: _size,)),
            color: Colors.red[500],
            onPressed: _setRatingAsThree,
            iconSize: _size,
          ),
        ),
      ],
    );
```



Application State

```
class Product extends Model {
  final String name;
  final String description;
  final int price;
  final String image;
  int rating;

  Product(this.name, this.description, this.price, this.image, this.rating);
  factory Product.fromMap(Map<String, dynamic> json) {
    return Product(
      json['name'],
      json['description'],
      json['price'],
      json['image'],
      json['rating'],
    );
  }
  void updateRating(int myRating) {
    rating = myRating; notifyListeners();
  }
}
```

Scoped Model

- Single
- Multiple

Scoped Model

- Single

```
ScopedModel<Product>(
  model: item, child: AnyWidget()
)
```

- Multiple

```
ScopedModel<Product>(
  model: item1,
  child: ScopedModel<Product>(
    model: item2, child: AnyWidget(),
  ),
)
```

Scoped Model

- Single

```
ScopedModel<Product>(
  model: item, child: AnyWidget()
)
```

- Multiple

```
ScopedModel<Product>(
  model: item1,
  child: ScopedModel<Product>(
    model: item2, child: AnyWidget(),
  ),
)
```

Usage:

```
ScopedModel.of<Product>(context).updateRating(2);
```

ScopedModelDescendant

- Content – ScopedModelDescendant pass the context of the application.
- Child – A part of UI, which does not change based on the model.
- Model – The actual model at that instance.

ScopedModelDescendant

- Context – ScopedModelDescendant pass the context of the application.
- Child – A part of UI, which does not change based on the model.
- Model – The actual model at that instance.

```
return ScopedModelDescendant<ProductModel>(
  builder: (context, child, model) => { ... Actual UI ... },
  child: StaticPartOfTheUI(),
);
```

Navigation

```
MaterialPageRoute(builder: (context) => Widget())
```

Navigation

```
MaterialPageRoute(builder: (context) => Widget())
```

```
Navigator.push( context, MaterialPageRoute(builder: (context) => Widget()), );
```

```
Navigator.pop(context);
```

Navigation

```
class MyHomePage extends StatelessWidget {
  MyHomePage({Key key, this.title}) : super(key: key);
  final String title;
  final items = Product.getProducts();

  @override
  Widget build(BuildContext context) {
    return Scaffold( appBar: AppBar(title: Text("Product Navigation")),
    body: ListView.builder(
      itemCount: items.length,
      itemBuilder: (context, index) {
        return GestureDetector(
          child: ProductBox(item: items[index]),
          onTap: () {
            Navigator.push(
              context, MaterialPageRoute(
                builder: (context) => ProductPage(item: items[index]),
              ),
            );
          },
        );
      },
    ));
  }
}
```

Navigation with Routes

```
MaterialApp(
  title: 'Named Routes Demo',
  // Start the app with the "/" named route. In this case, the app starts
  // on the FirstScreen widget.
  initialRoute: '/',
  routes: {
    // When navigating to the "/" route, build the FirstScreen widget.
    '/': (context) => const FirstScreen(),
    // When navigating to the "/second" route, build the SecondScreen widget.
    '/second': (context) => const SecondScreen(),
  },
)
```

Navigation to Route


```
// Within the `FirstScreen` widget
onPressed: () {
  // Navigate to the second screen using a named route.
  Navigator.pushNamed(context, '/second');
}
```

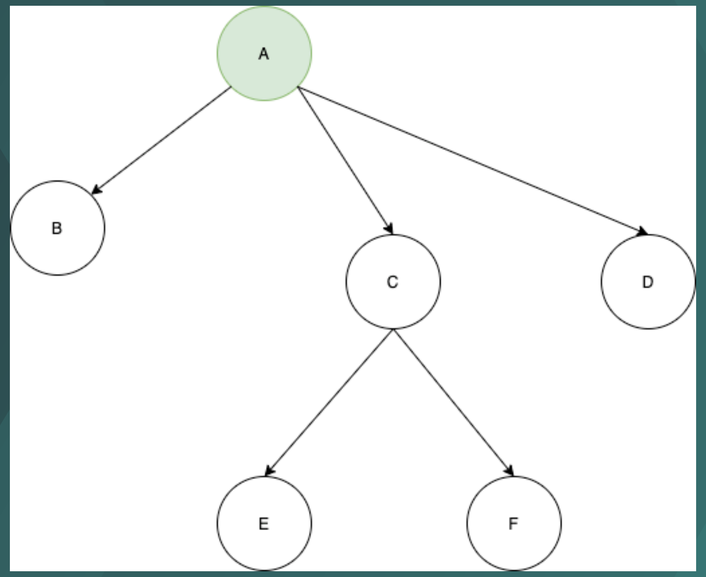
InheritedWidget

```dart
class FrogColor extends InheritedWidget {
  const FrogColor({super.key, required this.color, required super.child});

  final Color color;

  static FrogColor of(BuildContext context) {
    final FrogColor? result = context.dependOnInheritedWidgetOfExactType<FrogColor>();
    assert(result != null, 'No FrogColor found in context');
    return result!;
  }

  @override
  bool updateShouldNotify(FrogColor old) => color != old.color;
}
```

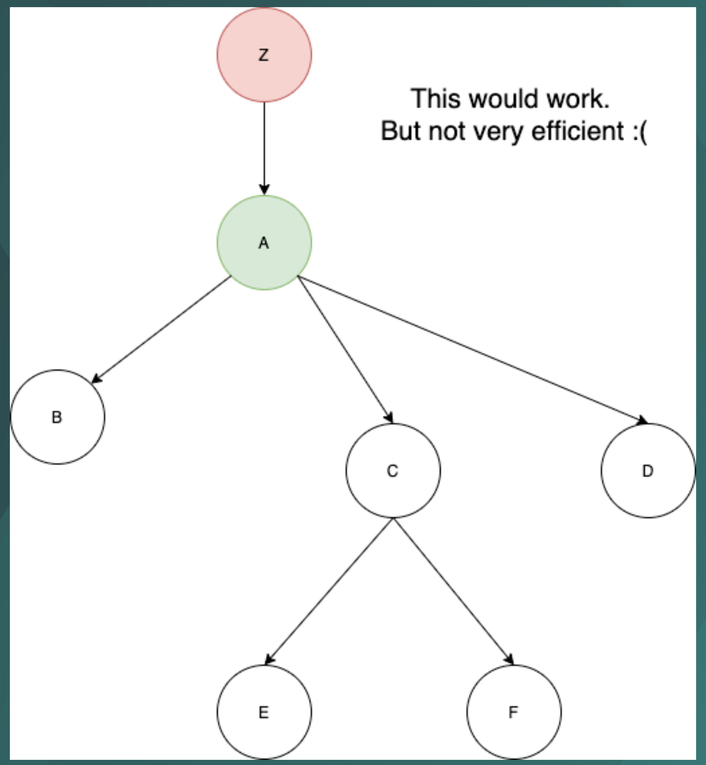
InheritedWidget

```
class MyPage extends StatelessWidget {
  const MyPage({super.key});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: FrogColor(
        color: Colors.green,
        child: Builder(
          builder: (BuildContext innerContext) {
            return Text(
              'Hello Frog',
              style: TextStyle(color: FrogColor.of(innerContext).color),
            );
          },
        ),
      ),
    );  }}
```

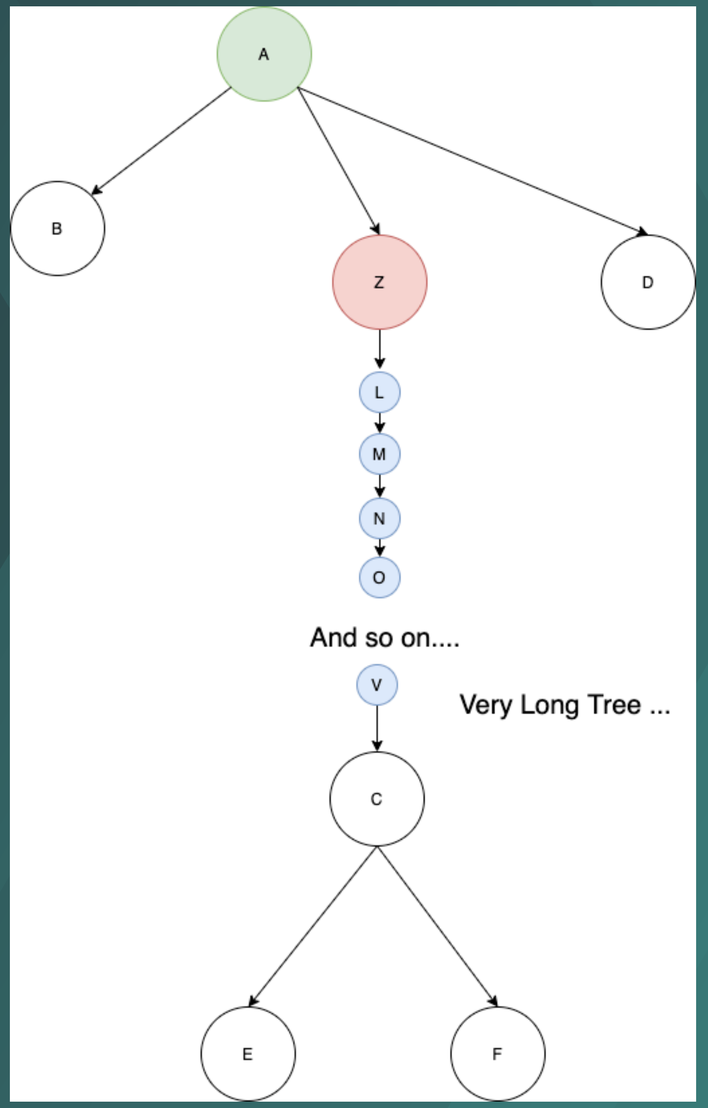
InheritedWidget



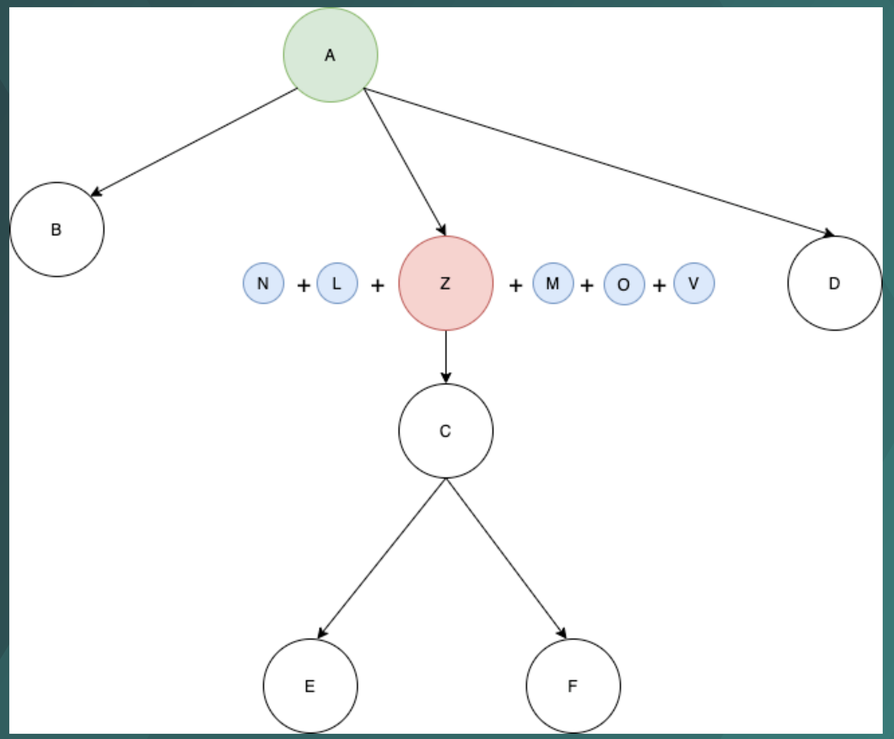
InheritedWidget



InheritedWidget



InheritedWidget



MultiProvider

```
dependencies:
  flutter:
    sdk: flutter

  # The following adds the Cupertino Icons font to your application.
  # Use with the CupertinoIcons class for iOS style icons.
  cupertino_icons: ^<version>
  provider: ^<version>
```

MultiProvider

```
dependencies:
  flutter:
    sdk: flutter

  # The following adds the Cupertino Icons font to your application.
  # Use with the CupertinoIcons class for iOS style icons.
  cupertino_icons: ^1.0.2
  provider: ^6.0.4
```

https://pub.dev/packages/provider

MultiProvider

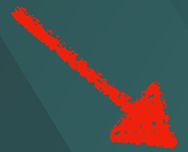
```
class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MultiProvider(
      providers: [
        ChangeNotifierProvider.value(
          value: Counter(),
        ),
      ],
      child: MaterialApp(
        title: 'Flutter Demo',
        theme: ThemeData(
          primarySwatch: Colors.blue,
        ),
        home: MyHomePage(title: "Provider Pattern"),
      ),
    );}
}
```

MultiProvider

```dart
class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MultiProvider(
      providers: [
        ChangeNotifierProvider.value(
          value: Counter(),
        ),
      ],
      child: MaterialApp(
        title: 'Flutter Demo',
        theme: ThemeData(
          primarySwatch: Colors.blue,
        ),
        home: MyHomePage(title: "Provider Pattern"),
      ),
    );}
}
```

```dart
class Counter extends ChangeNotifier {
  var _count = 0;

  int get getCounter {
    return _count;
  }

  void incrementCounter() {
    _count += 1;
    notifyListeners();
  }
}
```

MultiProvider



```
var counter = Provider.of<Counter>(context).getCounter;
```

MultiProvider


```
var counter = Provider.of<Counter>(context).getCounter;
```

```
void _incrementCounter(BuildContext context) {
  Provider.of<Counter>(context, listen: false).incrementCounter();
}
```

Lecture outcomes

- Widgets.
- Gestures.
- State Management.