Skärmar, navigering

TDP028 Entreprenöriell programmering

LINKÖPINGS UNIVERSITET

Översikt

- Lite mer Flutter
- Enklare navigering
  - Rak navigeringstruktur
  - Skicka data
- Mer kompex navigering
  - Olika del-träd
  - go_router
  - Hantering av bakåt-knapp

Ta sig från en skärm till en annan

```
onPressed: () { Navigator.push(
    context,
    MaterialPageRoute(
        builder: (context) => const ShoppingCartPage(),
        ... );}
```

Parameter: Anonyma funktioner som kommer anropas vid rätt tillfälle

Funktioner i Flutter

Syntax

Funktioner (liknar Java)

```
int fibonacci(int n) {
    if (n == 0 || n == 1) {
        return n;
    else {
        return fibonacci(n - 1) + fibonacci(n - 2);
    }
}
```

Funktioner (liknar Java)

```
int add(int x, int y) {
    return x + y;
}

// skriven på en rad
int add(int x, int y) { return x + y; }
```

Valfria (optional/formal) parametrar

```
// lista av valfria parametrar som inte behöver
// namnges vid anrop
doSomething(String s1, [String opt = 'a', int n = 5]) {
  ...
}
```

- Måste anges i rätt ordning när man skickar argument i funktionsanrop

Namngivna valfria parametrar

```
// dict of parameters, with default values
void setFont(String fontFamily,
                {bool bold = true, // default value is true
                 bool? italic}) {    // default value is false
    …
}

// call
setFont('Helvetica', {bold: false})  // italic will be false
```

Namngivna valfria parametrar

Non-nullable!

```
// dict of parameters
void setFont({required String fontFamily,
              bool bold = true,
              bool? italic}) {  // automatic default value
    …
}

// call
setFont('Helvetica', {bold: false})  // italic will be false
```

Funktioner som kan skrivas på en rad

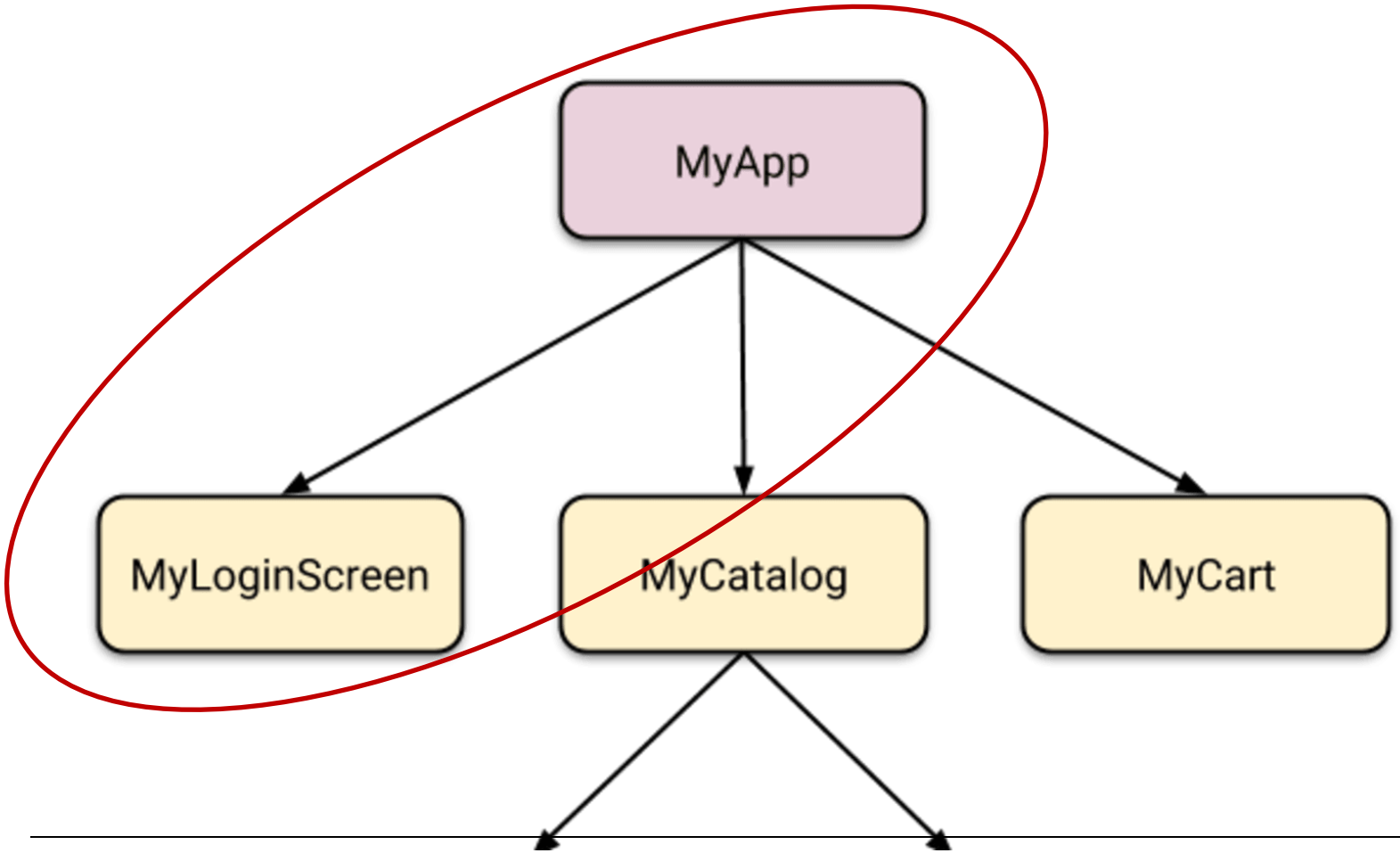
```
int add(int x, int y) {
    return x + y;
}

// skriven på en rad
int add(int x, int y) { return x + y; }
```

Arrow-syntax

```
fnc (arg) { return arg.contains('turn'); }  // vanlig syntax

fnc (arg) => arg.contains('turn')           // arrow-syntax

(arg) => arg.contains('turn')               // anonym funktion
```

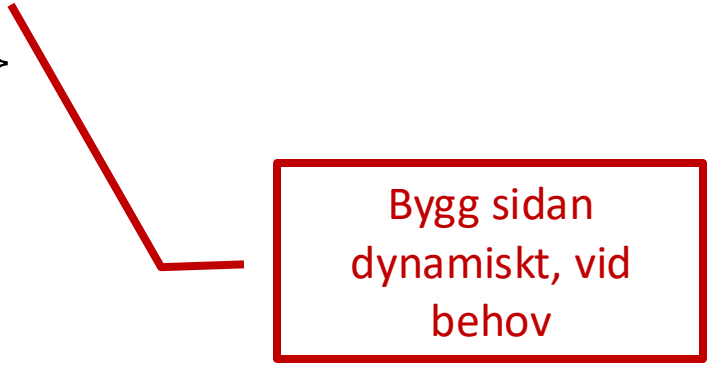
Widget-trädet

Vill bara bygga det som behövs vid navigering



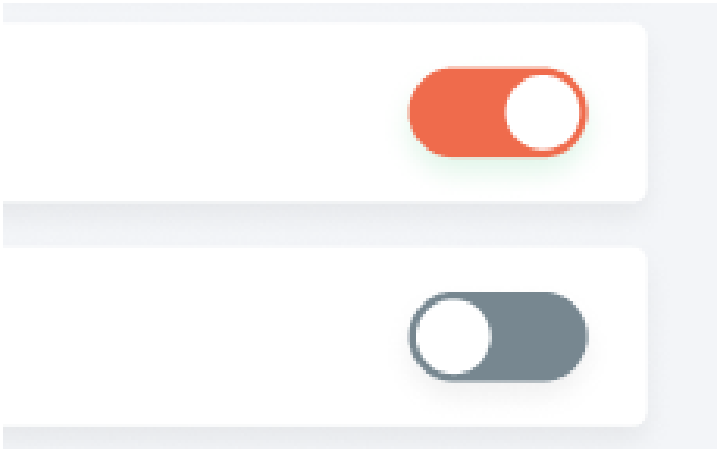
Ta sig från en skärm till en annan

```
onPressed: () { Navigator.push(
    context,
    MaterialPageRoute(
        builder: (context) => const ShoppingCartPage(),
        ... );}
```



Bygga Widgets på basis av tillstånd (state)

- Hur ska man bygga Widgets som ska visas olika
  - T.ex. beroende på om man tryckt på dem?



Två typer av Widgets

Stateless eller Stateful

StatelessWidget

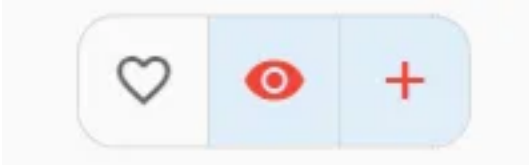
- Utan minne
  - Layout Widgets
    - Center
    - Column
  - Enklare widgets
    - Text

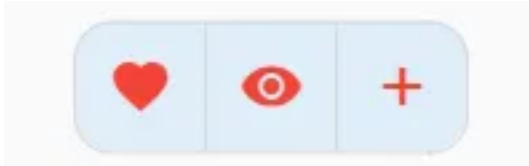
StatefulWidget

- Har internt minne (tillstånd)
- När widget:en behöver ritas upp på olika sätt beroende på vad som har hänt
  - Ex. gilla-knapp
    - Ifyllt hjärta när den har klickats

Exempel: Toggle button

```
ToggleButtons(
  ...
  onPressed: (index) {
    setState(() {
      isSelected[index] = !isSelected[index];
    });
  },
  children: [
    isSelected[0]
      ? Icon(Icons.favorite)
      : Icon(Icons.favorite_border),
```





Om state har förändrats

- Rita om allt som ligger under/inom scope för tillståndet som har förändrats
  - Måste få rätt information för Widget-bygget
    - Därför måste build ligga i State<MyWidget>
- Läs mer:
  - https://docs.flutter.dev/data-and-backend/state-mgmt/simple

Exempel: StatefulWidget

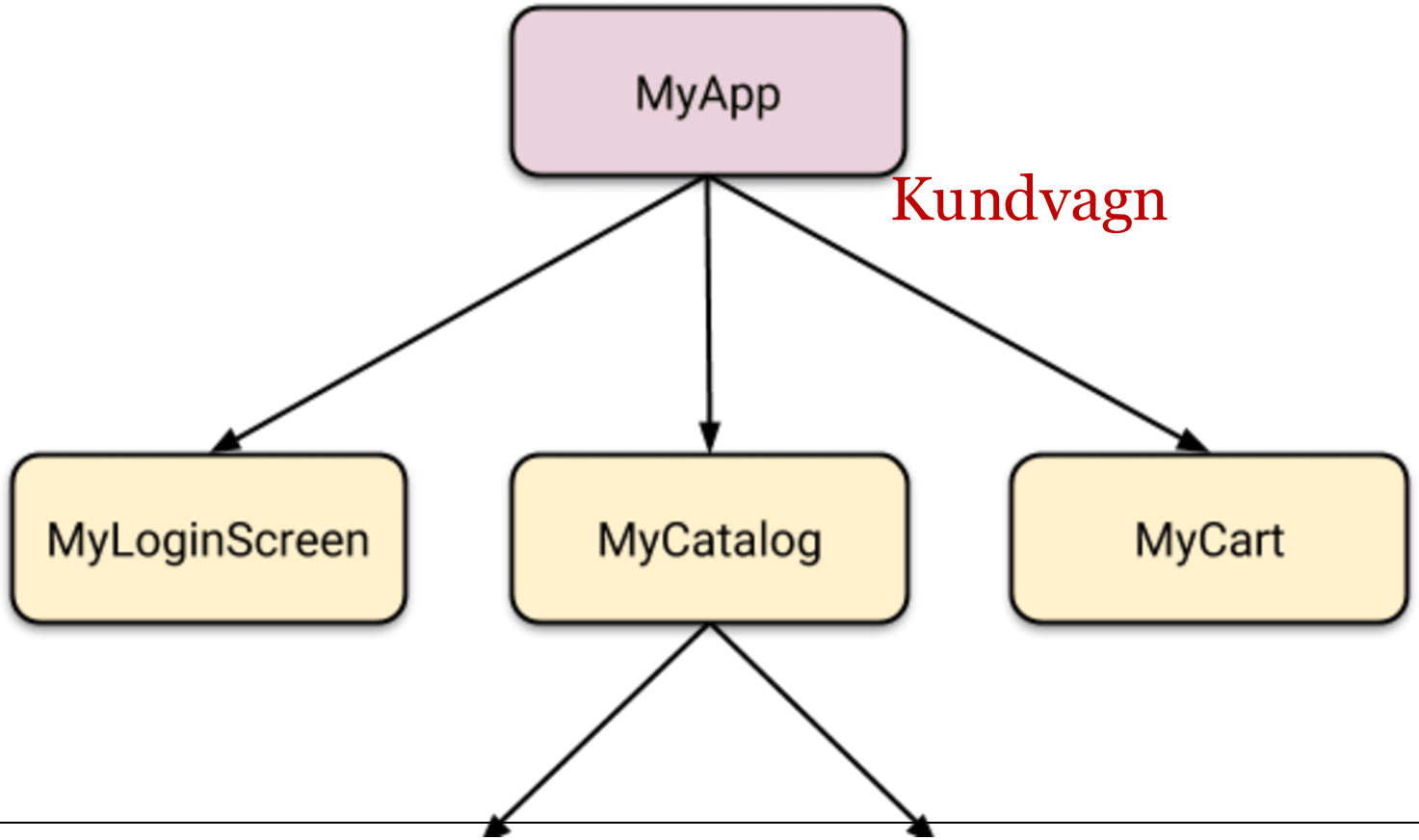
```
class Page extends StatefulWidget {
  @override
  State<Page> createState() => _PageState();
}

class _PageState extends State<Page> {
  var selectedIndex = 0;

  @override
  Widget build(BuildContext context) {
    switch (selectedIndex) { … // bygg vald Widget }
```

Globalt tillstånd: AppState

- Använd app state när minnet ska påverka olika widgets i olika delar av widget-trädet
  - T.ex. vilken användare som är inloggad i appen
  - T.ex. kundvagn i shopping-app

AppState – globalt tillstånd

När flera Widgets ska dela på tillstånd



Exempel på AppState

```
class MyAppState extends ChangeNotifier {
  var current = WordPair.random();

  void getNext() {
    current = WordPair.random();
    notifyListeners();
  }
```

Exempel på AppState

```
class MyAppState extends ChangeNotifier {
  var current = WordPair.random();

  void getNext() {
    current = WordPair.random();
    notifyListeners();
  }
```

Exempel: lyssna på AppState

```
class GeneratorPage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    var appState = context.watch<MyAppState>();
```

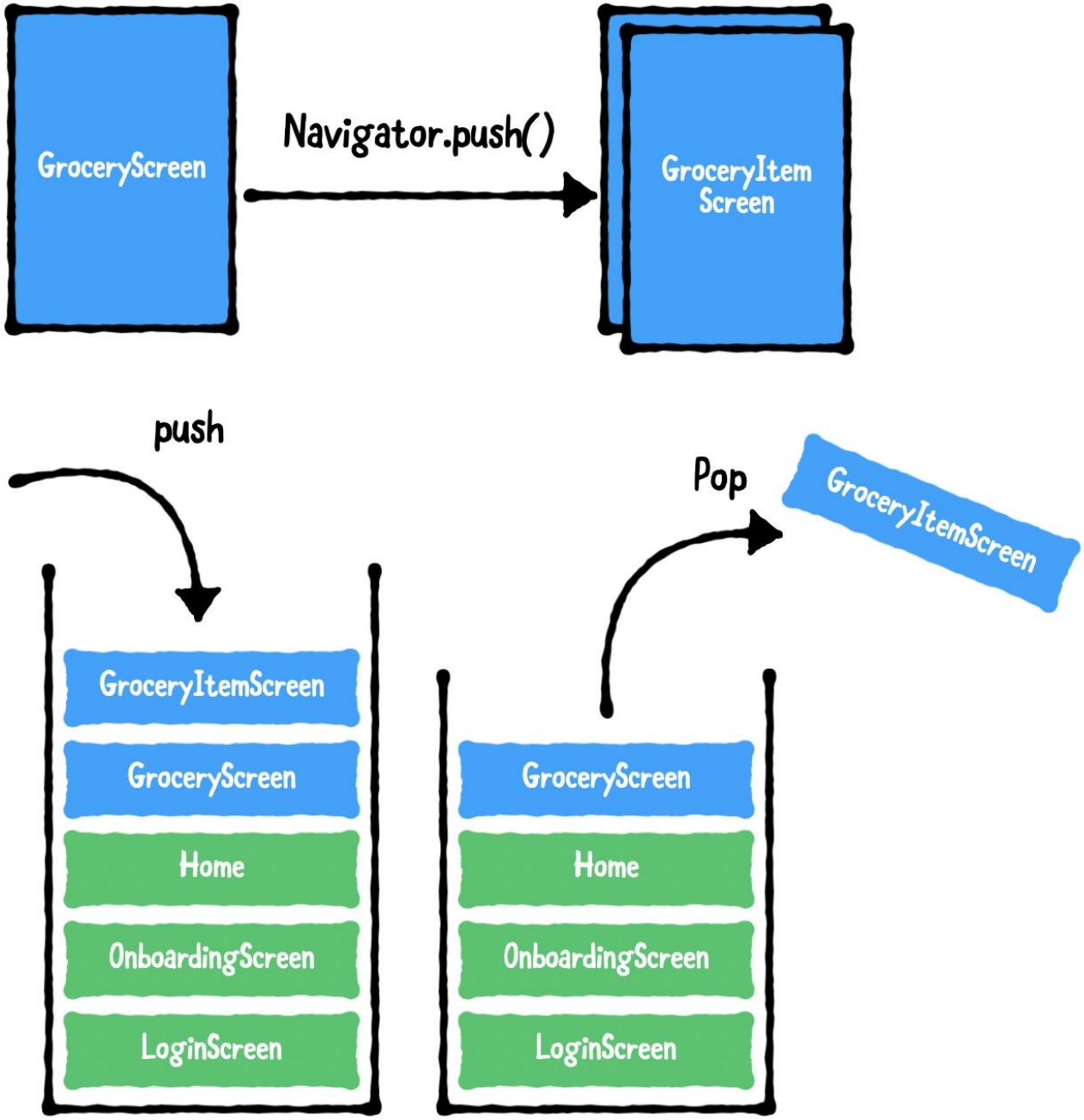
Vill observera ändringar i MyAppState

Navigering

Gå mellan skärmar

Navigator

- Kommer med Flutter
- Tillåter enkel navigering
  - Enkel = rak navigeringsstruktur
  - Backar upp samma väg man kom


GroceryScreen
Navigator.push()
GroceryItem Screen
push
GroceryItemScreen
GroceryScreen
Home
OnboardingScreen
LoginScreen
Pop
GroceryItemScreen
GroceryScreen
Home
OnboardingScreen
LoginScreen

Ta sig från en skärm till en annan

```
onPressed: () { Navigator.push(
    context,
    MaterialPageRoute(
      builder: (context) => const ShoppingCartPage(),
      … );}
```

Parameter: Anonym funktion som kommer anropas vid rätt tillfälle

- Men, bättre att använda go_router biblioteket
  - Kan då backa på rätt sätt

Skicka med argument

```
body: ListView.builder(
  itemCount: todos.length,
  itemBuilder: (context, index) {
    return ListTile(
      title: Text(todos[index].title),
      onTap: () {
        Navigator.push(
          context,
          MaterialPageRoute(
            builder: (context) => DetailsScreen(todo: todos[index])
```

Skicka med arg till konstruktorn för nästa skärm

Skicka information bakåt

```
child: ElevatedButton(
        onPressed: () {
          // Close the screen. Return ’Nope’ as result.
          Navigator.pop(context, 'Nope');
```

Skicka information bakåt

```
Future<void> _navToSelect (BuildContext context) async {
// Navigator.push returns a Future that completes after
// calling Navigator.pop on the Selection Screen
final result = await Navigator.push(
  context,
  MaterialPageRoute(
    builder: (context) => const SelectionScreen()), );
```

Läs mer:
https://docs.flutter.dev/cookbook/navigation/returning-data

Förhindra bakåt-navigering

```
class HomePage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return WillPopScope(
      onWillPop: () async => false,
      child: Scaffold(...)
```

Navigering med go_router

Installera go_router

$ flutter pub add go_router

- Kommandot ovan lägger följande i pubspec.yaml

```
dependencies:
  go_router: ^14.2.7  # använd senaste versionen
```

```
import 'package:go_router/go_router.dart';
```

- Länk: https://pub.dev/packages/go_router

Stegen

- Sätt upp en router med önskat navigationsträd
- Använd MaterialApp.router konstruktorn istf vanliga
- context.go(…) för att navigera

Exempel: Sätt upp trädet

```
final GoRouter _router = GoRouter(
  routes: <RouteBase>[
    GoRoute(
      path: '/',
      builder: (context, state) => FirstPage(),
      routes: <RouteBase>[
        GoRoute(
          path: ‘second’,
          builder: (context, state) => SecondPage(),
```

Ange routes i MaterialApp

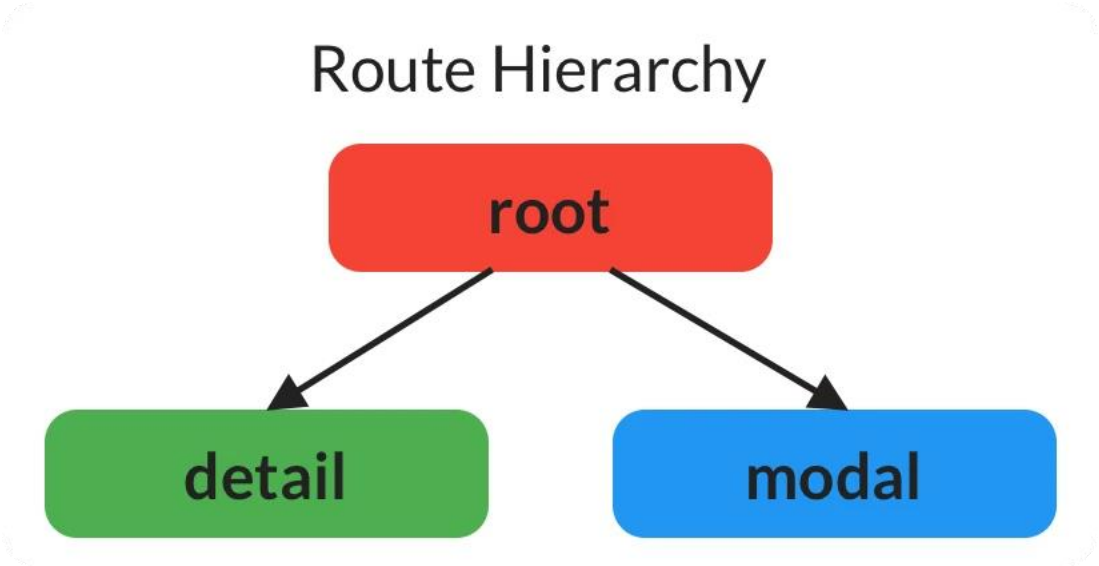
```
@override
Widget build(BuildContext context) {
  return MaterialApp.router(
    routerConfig: _appRouter,
      ...
```

Använd .router konstruktorn för MaterialApp

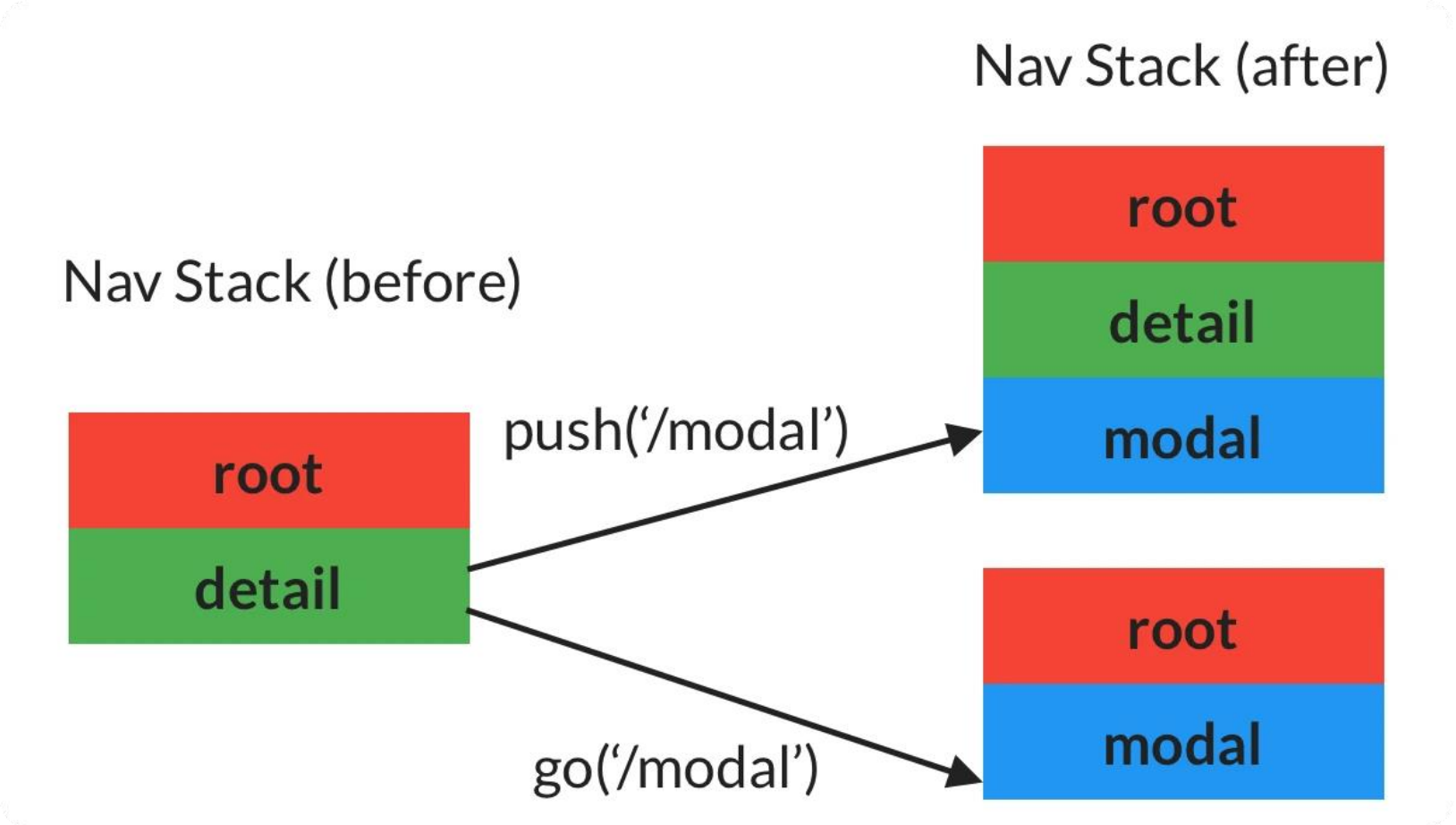
Utför navigeringen

```
return Scaffold(
    appBar: AppBar(
      leading: const Icon(Icons.menu),
      title: const Text('Shopping Mania'),
      actions: [
        IconButton(
          icon: const Icon(Icons.shopping_cart),
          onPressed: () => context.go('/cart'),
```

Skillnad mellan go() och push()



Skillnad mellan go() och push()



Skillnad context.go(…) – context.push(…)

- Läs mer på:
  - https://codewithandrea.com/articles/flutter-navigation-gorouter-go-vs-push/

”Magisk” tillgång till variabler…

```
actions: [
  IconButton(
    icon: const Icon(Icons.shopping_cart),
    onPressed: () => context.go('/cart')
  ),
  …
```

Dart är ett funktionellt språk

Funktioner är som vanliga objekt

Lambda-uttryck

- Funktioner som matematiska objekt
- $\lambda x. x + x^2$
  - $x + x^2$ som tar x som argument

Alonzo Church (1903-1995)

Inom datavetenskap

- Fördel med lambda-uttryck:
  - Kan lagras i variabler
  - Skickas som argument till andra funktioner

Lambda inom programmering

- Sätt att skriva namnlösa (anonyma) funktioner
  - Endast parametrar och funktionskropp
- I Java:
  - (parameter1, parameter2) -> { code block }
- I Flutter:
  - (parameter1, parameter2) => code block

Anonyma funktioner

```
// vi bryr oss inte om namnet på funktionen
(arg) => arg.contains('turn')

// kan skickas som argument till andra funktioner
someList.where(
                    (name) => name.contains('turn')
                  ).forEach(print);
```

Funktioner som objekt

```
void main() {
    // tilldela funktionsdef. till variabeln hej'
    var hej = (int x, int y) => x + y;

    print(hej(10, 20));
}
```

Lexikaliskt (statiskt) scope

- Fria (icke-lokala) variablers värden ”fryses” när funktionen *definieras*:

```
int x = 2;
add(y) => x + y;
x = 4;

add(3) ger 5   // ger inte 9
```

Closure i funktionell programmering

- Closure i förra exemplet:
  - {{x = 2},
    {add (y): x + y}}

- Fördel: Kan kompileras, eftersom vi vet alla variabelvärden

”Magisk” tillgång till variabler… closure

```
@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      actions: [
        IconButton(
          icon: const Icon(Icons.shopping_cart),
          onPressed: () => context.go('/cart’)
        ),
```

Anonym funktion, arrow-syntax

Navigering

Bakåt-knapp

Skicka parametrar framåt (go_router)

```
GoRoute(
  path: ’/users/:id',
  builder: (context, state) {
    // Get ’id’ param from URL that we are navigating to
    final id = state.pathParameters[’id’]!
    return UsersPage(userid: id);
  },
),
```

Förhindra bakåt-navigering (go_router)

- T.ex. ha login-skärmarna i separat del av nav-trädet
  - Hoppa till och från trädet med goNamed
    - Tar bort historik i förra delträdet

context.goNamed(‘/signin’)

…

context.goNamed(‘/page_after_signedin’)

Läs mer

- https://codewithandrea.com/articles/flutter-navigation-gorouter-go-vs-push/#:~:text=Think%20of%20go%20as%20a,of%20the%20existing%20navigation%20stack.

Klasser i Dart

Olika former av konstruktorer

```
@override
Widget build(BuildContext context) {
  return MaterialApp.router(
    routerConfig: _router,
  );
}
```

Konstruktor i Java

```
class Point {
   double x = 2.0;
   double y = 2.0;

   Point(x, y) {
      this.x = x;
      this.y = y;
}
```

Konstruktor i Flutter

```
class Point {
  double x = 2.0;
  double y = 2.0;

  // Sets the x and y instance variables
  // before the constructor body runs
  Point(this.x, this.y) {...}
}
```

Namngiven konstruktor i Flutter

```
const double xOrigin = 0;
const double yOrigin = 0;

class Point {
  final double x;
  final double y;

  Point(this.x, this.y);          // generativ konstruktor

  Point.atOrigin()                // namngiven konstruktor, ärvs inte nedåt
      : x = xOrigin,
        y = yOrigin;
}
```

Initialiseringslista

```
// Initializer list sets instance variables before
// the constructor body runs.
Point.fromJson(Map<String, double> json)
    : x = json['x']!,
      y = json['y']! {
  print(Initialized point coordinates to: ($x, $y)');
}
```

Kompilator-konstanta instanser

```
class ImmutablePoint {
  final double x, y;

  const ImmutablePoint(this.x, this.y);
}
```

Sätts bara en gång (under körning)

Kompilator-konstanta instanser

```
const ImmutablePoint p = ImmutablePoint(3, 6);

class ImmutablePoint {
    ...
    const ImmutablePoint(this.x, this.y);
}
```

Fabriker (factory-constructors)

```
class Logger {
  final String name;

  static final Map<String, Logger> _cache = <String, Logger>{};

  factory Logger(String name) {
    return _cache.putIfAbsent(name, () =>
        Logger._internal(name));
  }

  Logger._internal(this.name);
}
```

Dict av nycklar och Logger-instanser

Skapa ny instans om name **inte** finns i dict; lagra sedan i _cache

Fabriker (factory-constructors)

- I stället för konstruktor
  - När man vill kunna återanvända en instans av klassen
    - Ex. Logger ger samma instans för samma namn

- Läs mer på: https://dart.dev/language/constructors#factory-constructors

Generics

Begränsar/anger typen av element, argument, etc.

- Ex:

```
var names = <String>[];          // lista av strängar
```

- Ex:

```
T firstEl<T>(List<T> lotOfTs) {  // T bestäms vid anrop
    T element = ts[0];
    return element;
}
```

Generics

- Läs mer på:
  - https://dart.dev/language/generics

rita.kovordanyi@liu.se

www.liu.se

LINKÖPINGS UNIVERSITET