Experiment No 01

Aim: Create a simple flutter “hello World” application

Theory:

Flutter is an app SDK for building high-performance, high-fidelity apps for iOS, Android, Web(beta), and desktop(technical preview) from a single codebase written in Dart Language. Inthis article, I will provide line by line explanation of how to build a simple Hello World App using Flutter.

In Flutter everything is a Widget and using predefined widgets one can create user-defined widgets just like using int, float, double we can create user-defined data type. In Flutter there arethree types of widgets

- Stateless Widget
- Stateful Widget
- Inherited Widget

In this article, we will use Stateless Widget, Material App, Center, and Text Widget

Stateless Widget: In Flutter Stateless Widget are the widgets that can not change their state, thatis in Stateless Widget there is a method(function) called build which is responsible for drawing components on the screen called only once. To redraw a stateless widget one has to create a newinstance of the stateless widget.

MaterialApp: It is also a widget provided by Flutter Team, which follows Google Material Design Scheme, MaterialApp is a class that has various named arguments like home: in whichwe pass the widget that has to be displayed on Home Screen of an App. To read more
about MaterialApp check out Flutter Documentation.
Center Widget: Center is also a predefined widget by Flutter Team, which takes another widgetin its child argument. Using Center Widget as the name suggests it will display Widget in its child argument in Center.
Text Widget: The text widget is also predefined by Flutter Team, which is used to display text.Let us now build a Hello World App using Flutter.

Here we are importing the package which has a definition for Stateless Widget, Center, Text,Material App, and many more. It is like #include<iostream> in C++ program.

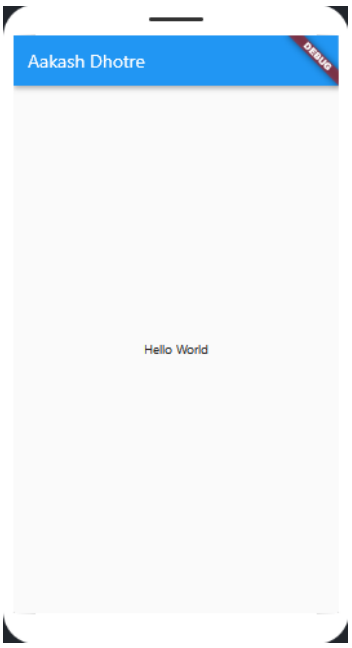
GeeksForGeeks: It is a user Defined class that inherits Stateless Widget, that is all the propertyof Stateless Widget is in GeeksForGeeks

Build: It is a method that is responsible for drawing components on the Screen it takes a BuildContext as an argument that has information about which widget has to be displayed andin which order it has to be painted on the screen.

Code:

```
import 'package:flutter/material.dart';

void main() {
  runApp(const HelloWorld());
}

class HelloWorld extends StatelessWidget {
  const HelloWorld({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    // Material App
    return MaterialApp(

        // Scaffold Widget
        home: Scaffold(
      appBar: AppBar(
        // AppBar takes a Text Widget
        // in it's title parameter
        title: const Text('Aakash Dhotre'),
      ),
      body: const Center(child: Text('Hello World')),
    ));
  }
}
```

Output:



Aakash Dhotre

Hello World

Conclusion:

Application (write two Application)

Questionnaire:

1. What is flutter?

2. Who developed the Flutter Framework and continues to maintain it today?

3. In Which programming language is used to build Flutter applications?

4. How many types of widgets are there in Flutter?

5. When building for iOS, Flutter is restricted to an________compilation strategy

6. A sequence of asynchronous Flutter events is known as a:

7. Access to a cloud database through Flutter is available through which service?

8. What are some key advantages of Flutter over alternate frameworks?

9. What element is used as an identifier for components when programming in Flutter?

10. What type of test can examine your code as a complete System?

Experiment No 02

Aim: create a simple application for increment and decrement counter using setstate

Theory: Import

First, we have the import statement. This imports the material.dart package which helps in implementing the Material Design throughout the app and for most of the Dart files that areused for UI design you have to import this package.

Main

Then you have the main function which has the runApp method containing the first Widget asits argument, in this case, it is called MyApp.

Stateless Widget (MyApp)

This widget is the root of the app. Here, inside the build method, it returns a Material Appwidget containing the three properties: title, theme & home.

Title

A one-line description used by the device to identify the app for the user.

On Android, the titles appear above the task manager's app snapshots which are displayedwhen the user presses the "recent apps" button. On iOS, this value cannot be used.

theme

This property defines the ThemeData widget containing information about different colors, fonts and other theme data that would be used throughout the app. Here, only one property isdefined, primarySwatch which stores the color blue. But, in your app, you can define any number of theme properties you want.

home

This property contains a Stateful Widget, MyHomePage to which the title is passed on. When you start this app for the first time this is the widget that will be displayed as the first screen onyour device.

Stateful Widget (MyHomePage)

This Stateful Widget MyHomePagetakes a constructor containing a key and a title. Here, the title is passed on from the previous class. You can see that the String title is marked final, this is done because the fields in a Widget subclass are always marked final if you do not mark thisas a final you will get a warning.

Now, we have the overridden createState method which returns the instance of the class_MyHomePageState.

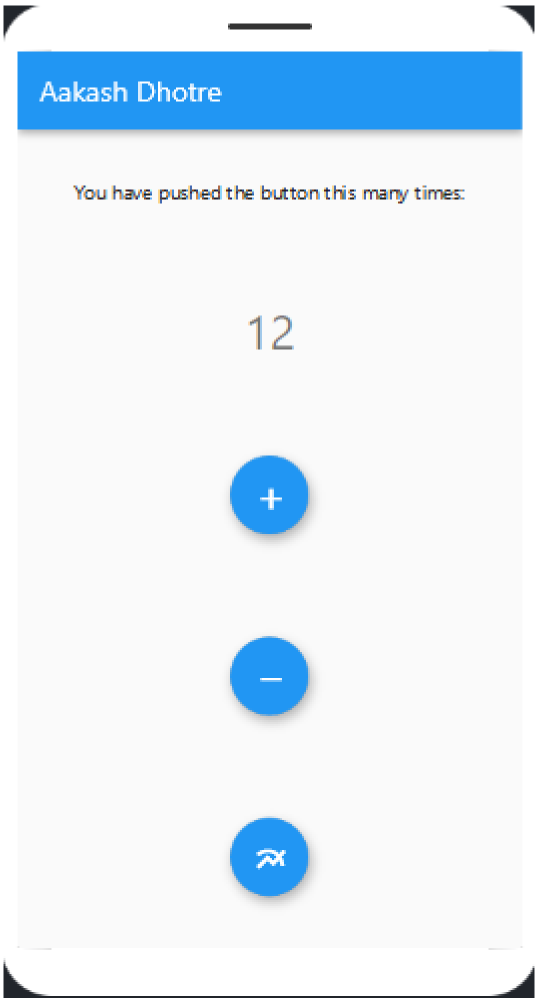
State class (_MyHomePageState)

In this class at first, a private integer variable _counter is initialized to zero. After that, we havea private void method defined called _incrementCounter.

Code:

```
// Copyright (c) 2019, the Dart project authors.  Please see the AUTHORS file
// for details. All rights reserved. Use of this source code is governed by a
// BSD-style license that can be found in the LICENSE file.

import 'package:flutter/material.dart';

void main() => runApp(MyApp());

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      debugShowCheckedModeBanner: false,
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: const MyHomePage(title: 'Aakash Dhotre'),
    );
  }
}

class MyHomePage extends StatefulWidget {
  final String title;

  const MyHomePage({
    Key? key,
    required this.title,
  }) : super(key: key);

  @override
  State<MyHomePage> createState() => _MyHomePageState();
}

class _MyHomePageState extends State<MyHomePage> {
  int _counter = 0;

  void _incrementCounter() {
    setState(() {
      _counter++;
    });
  }

  void _decrementCounter() {
    setState(() {
      _counter--;
    });
  }

  void _multiplyby2Counter() {
    setState(() {
      _counter = _counter * 2;
    });
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text(widget.title),
      ),
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.spaceAround,
          children: [
            const Text(
              'You have pushed the button this many times:',
            ),
            Text(
              '$_counter',
              style: Theme.of(context).textTheme.headlineMedium,
            ),
            FloatingActionButton(
              onPressed: _incrementCounter,
              tooltip: 'Increment',
              child: const Icon(Icons.add),
            ),
            FloatingActionButton(
              onPressed: _decrementCounter,
              tooltip: 'decrement',
              child: const Icon(Icons.remove),
            ),
            FloatingActionButton(
              onPressed: _multiplyby2Counter,
              tooltip: 'multiply',
              child: const Icon(Icons.multiline_chart_outlined),
            ),
          ],
        ),
      ),
    );
  }
}
```

Output:


Aakash Dhotre

You have pushed the button this many times:

12

Conclusion:

Application (write two Application)

Questionnaire:

1. What type of Flutter animation allows you to represent real-world behavior?

2. True or false Flutter boasts improved runtime performance over most applicationframeworks.

3. What command would you use to compile your Flutter app in release mode?

4. Which function will return the widgets attached to the screen as a root of the widget treeto be rendered on screen?

5. What is the key configuration file used when building a Flutter project?

6. A True or false: an experienced Flutter developer doesn't need to know platform nativelanguages or tools to build apps.

7. Which component allows us to specify the distance between widgets on the screen?

8. Which widget type allows you to modify its appearance dynamically according to user input?

9. What command would you run to verify your Flutter install and ensure your environment isset up correctly?

10. Which release mode will not contain any debugging data when run?

Experiment No 03

Aim: create application for Form and Form Validator

Theory:

Apps often require users to enter information into a text field. For example, you might requireusers to log in with an email address and password combination.

To make apps secure and easy to use, check whether the information the user has provided isvalid. If the user has correctly filled out the form, process the information. If the user submitsincorrect information, display a friendly error message letting them know what went wrong.

In this example, learn how to add validation to a form that has a single text field using the following steps:

1. Create a Form with a GlobalKey.
2. Add a TextFormField with validation logic.
3. Create a button to validate and submit the form.

First, create a Form. The Form widget acts as a container for grouping and validatingmultiple form fields.

When creating the form, provide a GlobalKey. This uniquely identifies the Form, and allowsvalidation of the form in a later step.

Add a TextFormField with validation logic

Although the Form is in place, it doesn’t have a way for users to enter text. That’s the job of a TextFormField. The TextFormField widget renders a material design text field and can displayvalidation errors when they occur.

Validate the input by providing a validator() function to the TextFormField. If the user’s inputisn’t valid, the validator function returns a String containing an error message. If there are no errors, the validator must return null.

For this example, create a validator that ensures the TextFormField isn’t empty. If it is empty,return a friendly error message.

Now that you have a form with a text field, provide a button that the user can tap to submit theinformation.

When the user attempts to submit the form, check if the form is valid. If it is, display a successmessage. If it isn’t (the text field has no content) display the error message.

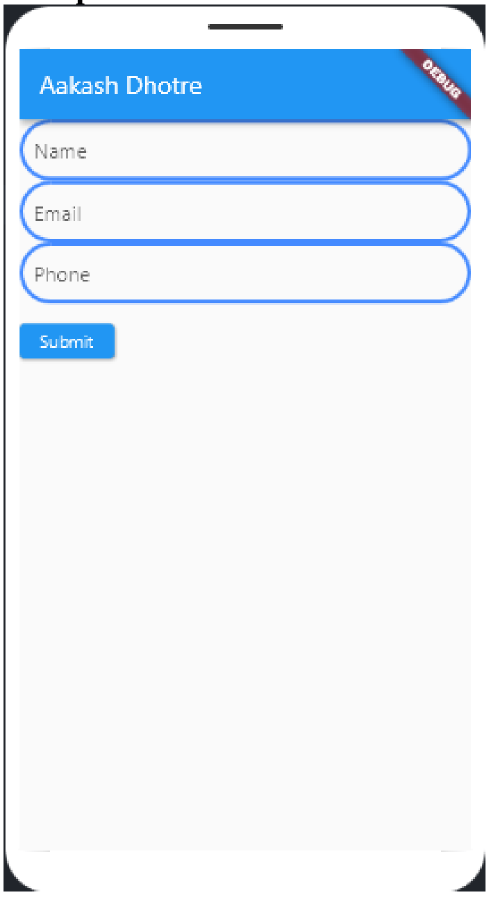
Code:

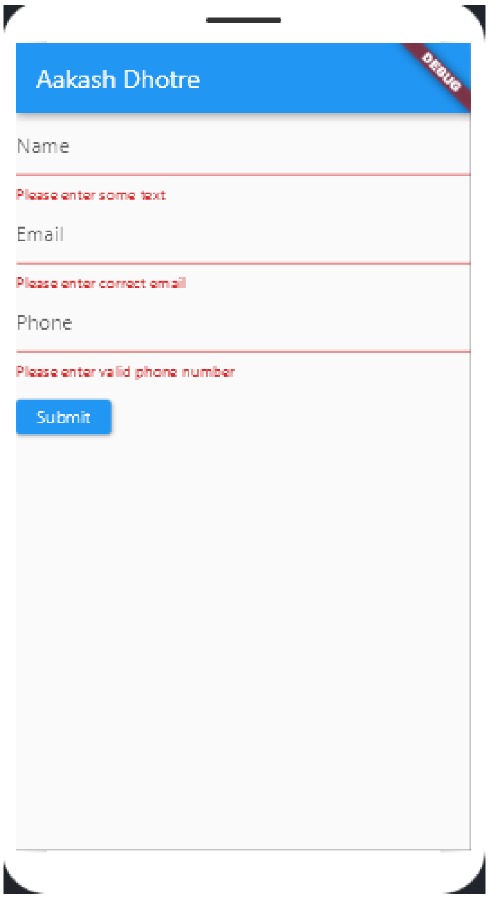
```
import 'package:flutter/material.dart';

void main() => runApp(const MyApp());
```

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

  @override
  Widget build(BuildContext context) {
    const appTitle = 'Aakash Dhotre';

    return MaterialApp(
      title: appTitle,
      home: Scaffold(
        appBar: AppBar(
          title: const Text(appTitle),
        ),
        body: const MyCustomForm(),
      ),
    );
  }
}

// Create a Form widget.
class MyCustomForm extends StatefulWidget {
  const MyCustomForm({super.key});

  @override
  MyCustomFormState createState() {
    return MyCustomFormState();
  }
}

// Create a corresponding State class.
// This class holds data related to the form.
class MyCustomFormState extends State<MyCustomForm> {
  // Create a global key that uniquely identifies the Form widget
  // and allows validation of the form.
  //
  // Note: This is a GlobalKey<FormState>,
  // not a GlobalKey<MyCustomFormState>.
  final _formKey = GlobalKey<FormState>();

  @override
  Widget build(BuildContext context) {
    // Build a Form widget using the _formKey created above.
    return Form(
      key: _formKey,
      child: Column(
        crossAxisAlignment: CrossAxisAlignment.start,
        children: [
          TextFormField(
            decoration: InputDecoration(
          enabledBorder: OutlineInputBorder(
            borderSide: BorderSide(width: 3, color: Colors.blueAccent),
            borderRadius: BorderRadius.circular(50.0)),
          hintText: 'Enter your full name',
          labelText: 'Name',
         ),
         // The validator receives the text that the user has entered.
         validator: (value) {
          if (value == null || value.isEmpty) {
           return 'Please enter some text';
          }
          return null;
         },
        ),
        TextFormField(
         decoration: InputDecoration(
          enabledBorder: OutlineInputBorder(
            borderSide: BorderSide(width: 3, color: Colors.blueAccent),
            borderRadius: BorderRadius.circular(50.0)),
          hintText: 'Enter your email',
          labelText: 'Email',
         ),
         validator: (value) {
          if (value == null || value.isEmpty) {
           return 'Please enter correct email';
          }
          if (value == null ||
            value.isEmpty ||
            !RegExp(r"^[a-zA-Z0-9.a-zA-Z0-9.!#$%&'*+-/=?^_`{|}~]+@[a-zA-Z0-9]+\.[a-
   zA-Z]+")
              .hasMatch(value)) {
           return 'Enter a valid email!';
          }
          return null;
         },
        ),
        TextFormField(
         decoration: InputDecoration(
          enabledBorder: OutlineInputBorder(
            borderSide: BorderSide(width: 3, color: Colors.blueAccent),
            borderRadius: BorderRadius.circular(50.0)),
          hintText: 'Enter a phone number',
          labelText: 'Phone',
         ),
         validator: (value) {
          if (value == null ||
            value.isEmpty ||
            !RegExp(r'(^(?:[+0]9)?[0-9]{10,12}$)').hasMatch(value)) {
           return 'Please enter valid phone number';
          }
```

```
              return null;
            },
          ),
          Padding(
            padding: const EdgeInsets.symmetric(vertical: 16.0),
            child: ElevatedButton(
              onPressed: () {
                // Validate returns true if the form is valid, or false otherwise.
                if (_formKey.currentState!.validate()) {
                  // If the form is valid, display a snackbar. In the real world,
                  // you'd often call a server or save the information in a database.
                  ScaffoldMessenger.of(context).showSnackBar(
                    const SnackBar(content: Text('Processing Data')),
                  );
                }
              },
              child: const Text('Submit'),
            ),
          ),
        ],
      ),
    );
  }
}
```

Output:





Please enter some text

Please enter correct email

Please enter valid phone number

Conclusion:

Application (write two Application)

Questionnaire:

1. What language is Flutter's rendering engine primarily written in?

2. What is a drawback of Flutter that might lead you to choose another solution?

3. True or False: Flutter supports desktop application development.

4. What widget would you use for repeating content in Flutter?

5. True or False: Flutter teams are inherently more difficult to manage because the framework isso new.

6. Flutter is mainly optimized for 2D mobile apps that can run on?

7. It is very necessary to learn Dart language for building Flutter application?

8. A widget that allows us to refresh the screen is called a_______________.

9. The examples of the Stateless widget are?

10. pubspec.yaml file does contain?

Experiment No 04

Aim: Create application that illustrate State management using bloc and provider

Theory:

The provider package is an easy-to-use package which is basically a wrapper around the InheritedWidgets that makes it easier to use and manage. It provides a state management technique that is used for managing a piece of data around the app.

The basic classes available in the provider package are –

ChangeNotifierProvider<T extends ChangeNotifier> It listens to a ChangeNotifier extended bythe model class, exposes it to its children and descendants, and rebuilds depends
whenever notifyListeners is called.

```
ChangeNotifierProvider(

  create: (context) =>

  DataModel(),child: ...

)
```

Consumer<T> It obtains the provider from its ancestors and passes the value obtained to thebuilder.

```
@override

 Widget build(BuildContext

   context) {return

   Consumer<DataModel>(

     builder: (context, data, child) => DataWidget(par1: par1, par2:

     par2),child: Text(data.first),

   );

 }
```

FutureProvider<T> This class listens for a Future and then passes its values to its children anddescendants.
Constructors

```
FutureProvider<T>(
    {Key key,
    @required Create<Future<T>>
    create,T initialData,
    ErrorBuilder<T> catchError,
    UpdateShouldNotify<T>
    updateShouldNotify,bool lazy,
    TransitionBuilder
    builder,Widget child}
```

)

What is flutter bloc?

Flutter bloc is one of the state ma nagement for Flutter applications. You can use it to handle all

the possible states of your application in an easy way.

Flutter bloc is simple to use because you and your team are going to understand the concept

quickly, no matter what is your level, this library has very good documentation with tons of

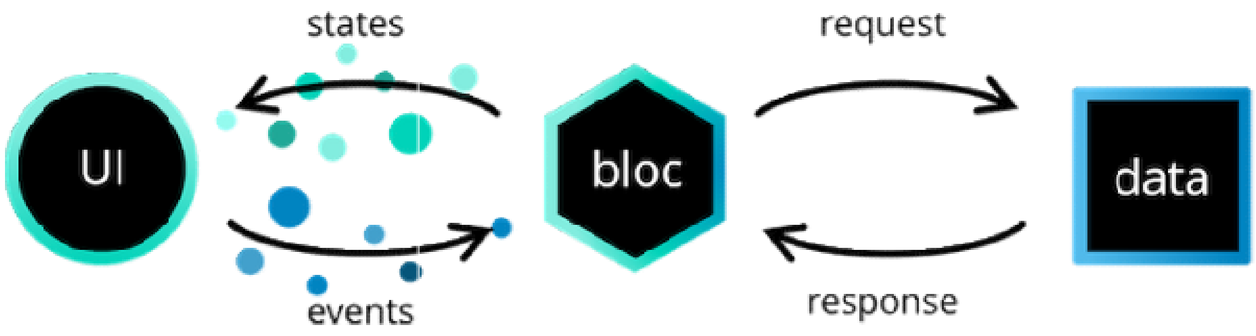
examples and also, is one of the most used in the flutter community, so if you ha e any question

or problem you probably are going to find the solution with a simple search on the internet.

Is powerful because is going to help you to create all kinds of applications, you can create

applications for learning purposes for example, and also you can create complex

applications in aproduction environment and flutter bloc is valid in both cases.



Code:
Using Provider:
main.dart

```
import 'dart:developer';

import 'package:flutter/material.dart';
import 'model.dart';
import "package:provider/provider.dart";

void main() {
  runApp(ChangeNotifierProvider(
      create: ((context) {
        return Counter_Model();
      }),
      child:const DemoPage()))
  ;
}

class DemoPage extends StatelessWidget {
  const DemoPage();
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
        theme: ThemeData.from(colorScheme: const ColorScheme.light()),
        home:  const DemoPageChild());
  }
}

class DemoPageChild extends StatelessWidget {
  const DemoPageChild();

  @override
  Widget build(BuildContext context) {
    return Scaffold(
        body:const DemoPageChildWidget() );
  }
}
class DemoPageChildWidget extends StatelessWidget{
  const DemoPageChildWidget();

  @override
  Widget build(BuildContext context) {
    return Padding(
        padding: EdgeInsets.only(top: 15.0),
        child: Row(children: [
          Expanded(
              child: ElevatedButton(
                onPressed: () {
                  context.read<Counter_Model>().increament();
                },
                child: const Text("+", textScaleFactor: 3),
              )),
          Expanded(
              child: Center(
                  child: const ShowCounter()




              )
          ),
          Expanded(
              child: ElevatedButton(
                onPressed: () {
                  context.read<Counter_Model>()
                      .decrement();
                },
                child: const Text(
                  "-",
                  textScaleFactor: 3,
                )))
          ]));
    }




}
class ShowCounter extends StatelessWidget{
  const ShowCounter();
  @override
  Widget build(BuildContext context) {
    return   Text(
        context.watch<Counter_Model>()
            .counter
            .toString(),
        textScaleFactor: 3);


  }}
```

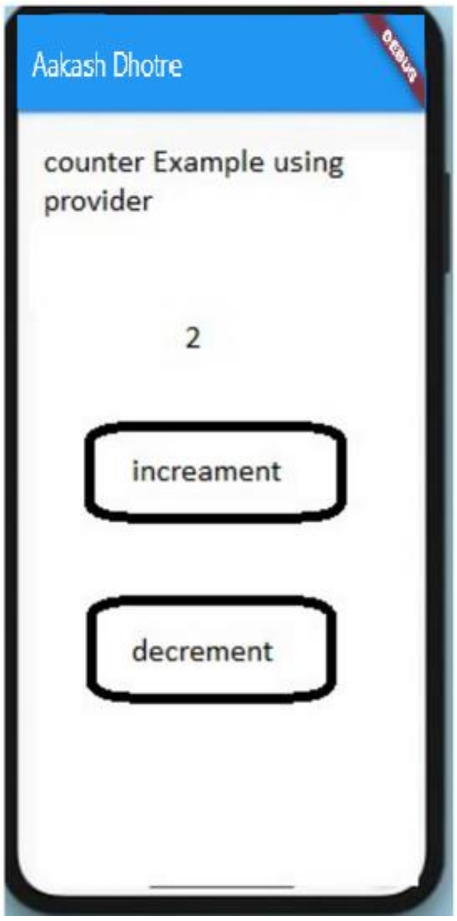
model.dart

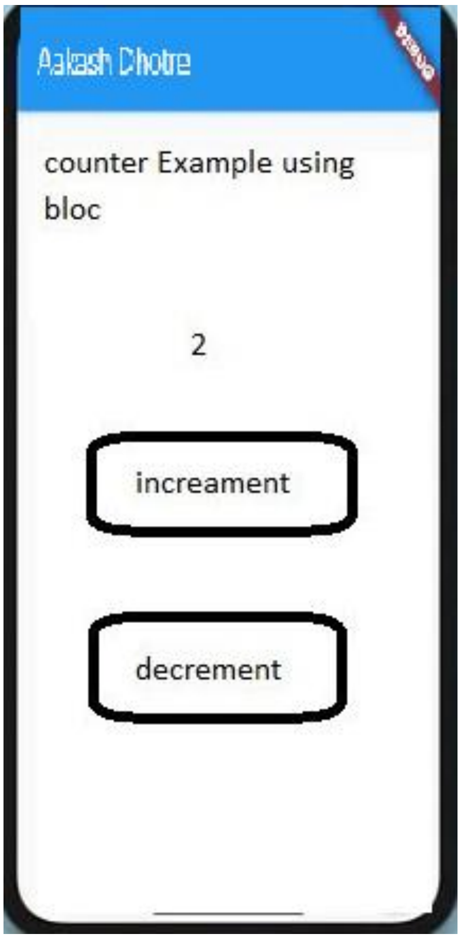
```
import 'package:flutter/material.dart';

class Counter_Model with ChangeNotifier{
  int _counter=0;
  int get counter=>_counter;
  void increament()
  {
    _counter++;
    notifyListeners();

  }
  void decrement()
  {
    _counter--;
    notifyListeners();

  }
}
```

Using bloc:

```
import 'package:flutter/material.dart';
import 'bloc.dart';

void main() {
  runApp(BlocProvider<CounterBloc>(create: (context)=>CounterBloc(),child:MaterialApp(
    debugShowCheckedModeBanner: false,
    title: "Bloc Counter",
    home: Scaffold(
      appBar: AppBar(title: Text("Bloc Counter")),
      body: counter_body(),
    ),
  )));
}

class counter_body extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Center(
        child:  Row(children: <Widget>[
          Expanded(
              child: ElevatedButton(
                  onPressed: () {
                    BlocProvider.of<CounterBloc>(context).add(Increment());
                    //below line also works instead of above line
                    //context.read<CounterBloc>().add(Increment());
                  },
                  child: Text(
                    "+",
                    textScaleFactor: 4,
                  ),
                  style: ElevatedButton.styleFrom(
                      fixedSize: const Size(240, 80)))),
          Expanded(
              child: Center(
                  child:BlocBuilder<CounterBloc,int>(builder: (context, count) {
                    return Text(
                      '$count',textScaleFactor: 4,
                      style: TextStyle(fontSize: 24.0),
                    );}







                  )
              )),
          Expanded(
              child: ElevatedButton(
                  onPressed: () {

                    BlocProvider.of<CounterBloc>(context).add(Decrement());
                    //below line also works instead of above line
                   // context.read<CounterBloc>().add(Decrement());


                    },
                  child: Text(
                    "-",
                    textScaleFactor: 4,
                  ),
                  style: ElevatedButton.styleFrom(
                      fixedSize: const Size(240, 80))))
        ]));
  }
}

abstract class CounterEvent {}

class Increment extends CounterEvent {}

class Decrement extends CounterEvent {}

class CounterBloc extends Bloc<CounterEvent, int> {
  void onChange(Change<int> change) //overriding function executed when there is a change
  {
    debugPrint("$change");
    super.onChange(change);
  }

  CounterBloc() : super(0) {
    on<Increment>((event, emit) => emit(state + 1));
    on<Decrement>((event, emit) => emit(state - 1));
  }
}
```

Output:





Conclusion:

Application (write two Application)

Questionnaire:

1. What widget would you use for repeating content in Flutter?

2. What language is Flutter's rendering engine primarily written in?

3. Which component allows us to specify the distance between widgets on the screen?

4 What type of test can examine your code as a complete system?

5. What element is used as an identifier for components when programming in Flutter?

6. Access to a cloud database through Flutter is available through which service?

7. What are the type of tests, that we can perform in flutter?

8 What does SDK stands for?

9. Which is used to load images from the flutter project's assets?

10. Everything is a widget in Flutter. True or False?

Experiment No 05

Aim: create application that illustrate animation in flutter

Theory:

Animation Controllers

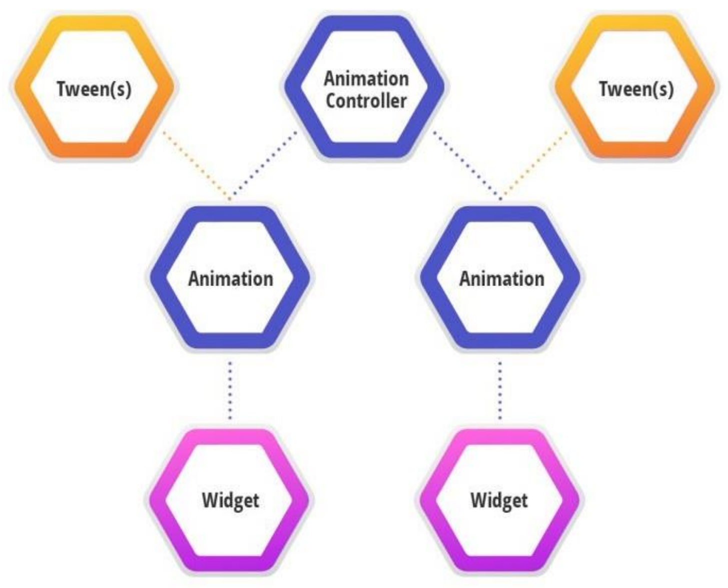
An animation controller is, as the name suggests, a way to control (trigger, fling or stop) ananimation.

However, the main function of the controller is to drive the animation. This means it will causethe animation to change its value and return a new value from the Tween based on the progression of the Animation.

Every Flutter animation needs at least two elements to be created:

- A Tween to get generate values for the animation
- An AnimationController as parent

An AnimationController gives the progression of the animation from 0 to 1 whereas theAnimation gives the actual tween value expected by the widget.



Animation controllers need to be disposed off once the navigation is complete.

The AnimationController also gives us control over how the animation behaves. For example,we can repeat an animation using *controller.repeat()*.

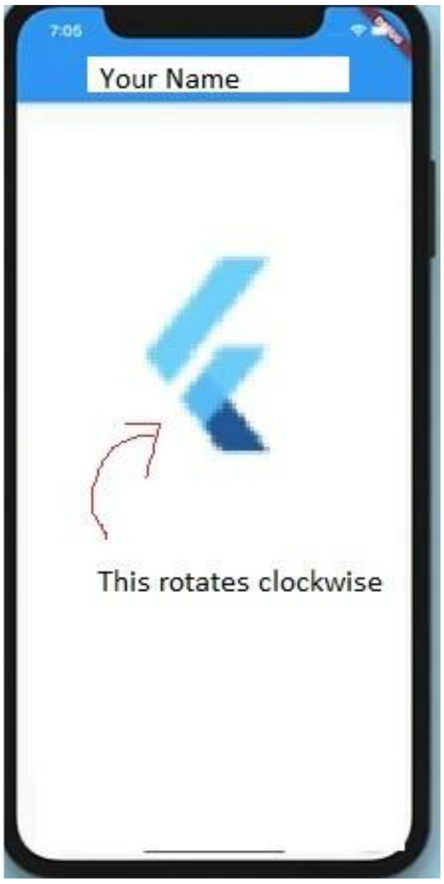
Note, however, that the Flutter animations we create are not tied to a specific object but rather,any widget in the hierarchy can use it as the need be.

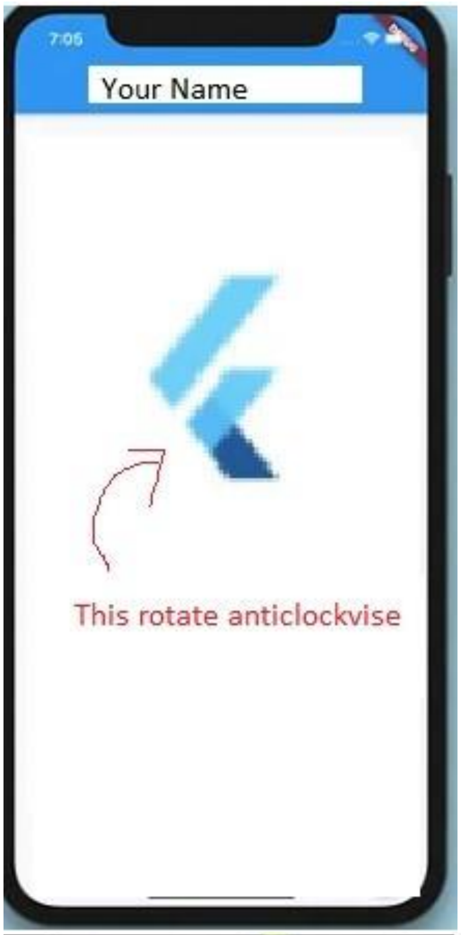
That’s it! You now have a good understanding of the basic components. Let’s dive right in tocreating our own animation then!

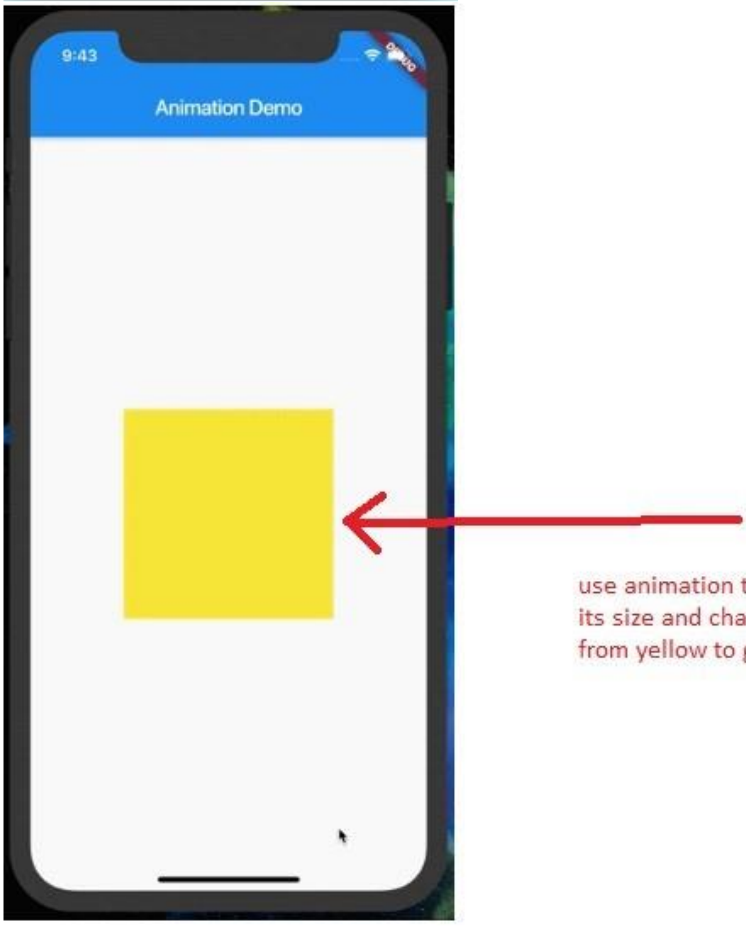
Code:

```
import 'dart:math';

import 'package:flutter/material.dart';

void main() {
  runApp(MaterialApp(
    home: Scaffold(
      body: Center(child: tween_color()),
    ),
  ));
}

class tween_color extends StatefulWidget {
  @override
  State<StatefulWidget> createState() {
    return tween_colorState();
  }
}

class tween_colorState extends State<tween_color>
    with TickerProviderStateMixin {
  late final AnimationController _controller;
  late final Animation<Color> _tween_color;

  void initState() {
    super.initState();
    _controller =
        AnimationController(duration: Duration(seconds: 9), vsync: this)
          ..reverse();
    _tween_color = Tween<Color>(begin: Colors.black, end: Colors.amberAccent)
        .animate(_controller);
  }

  void dispose() {
    _controller.dispose();
    super.dispose();
  }

  @override
  Widget build(BuildContext context) {
    int i = 0xff4caf50;
    return AnimatedBuilder(
      animation: _controller,
      builder: (_, __) {
        debugPrint("${_tween_color.value}");
        return ElevatedButton(
          child: const Text("Animate me!"),
          style: ElevatedButton.styleFrom(
            primary: _tween_color.value,
          ), //
          onPressed: () {
            if (_controller.status == AnimationStatus.dismissed) {
              _controller.forward();

              // debugPrint("if");
            } else {
              //debugPrint("else ${_controller.status}");
              _controller.reverse();
            }

            // Color(0xfff44336))" and "MaterialColor(primary value: Color(0xff4caf50))"
          },
        );
      },
    );
  }
}
```

Output:







use animation to resize
its size and change color
from yellow to greea

Conclusion:

Application (write two Application)

Questionnaire:

1. __ is the programming language used to code flutter apps and employ many of its moreadvanced features.

2. A _____ widget in Flutter is a box with a specific size.

3. What is the refresh rate of our animation in Flutter?

4. What are the subclasses of Key?

5. Which commands is used to compile release mode?

6. In Flutter, which is used as identifiers for widgets, elements, and semantic nodes?

7. What operator evaluates and returns the value between two expressions?

8. Which is a sequence of asynchronous events?

9. Which are the best editors of flutter development?

10. Which are the best editors of flutter development?

Experiment No 06

Aim: create application that illustrate explain routing, passing data using navigator and navigation rail

Theory:

This recipe uses the Navigator to navigate to a new route.
The next few sections show how to navigate between two routes, using these steps:

1. Create two routes.
2. Navigate to the second route using Navigator.push().
3. Return to the first route using Navigator.pop().

Create two routes
First, create two routes to work with. Since this is a basic example, each route contains only a single button. Tapping the button on the first route navigates to the second route. Tapping thebutton on the second route returns to the first route.

Navigate to the second route using Navigator.push()
To switch to a new route, use the Navigator.push() method. The push() method adds a Route to the stack of routes managed by the Navigator. Where does the Route come from? You can createyour own, or use a MaterialPageRoute, which is useful because it transitions to the new route using a platform-specific animation.
In the build() method of the FirstRoute widget, update the onPressed() callback:Return to the first route using Navigator.pop()

How do you close the second route and return to the first? By using the Navigator.pop() method.The pop() method removes the current Route from the stack of routes managed by the Navigator.
To implement a return to the original route, update the onPressed() callback inthe SecondRoute widget:

To work with named routes, use the Navigator.pushNamed() function. This example replicatesthe functionality from the original recipe, demonstrating how to use named routes using the following steps:

1. Create two screens.
2. Define the routes.
3. Navigate to the second screen using Navigator.pushNamed().
4. Return to the first screen using Navigator.pop().

Create two screens

First, create two screens to work with. The first screen contains a button that navigates to thesecond screen. The second screen contains a button that navigates back to the first.

Define the routes

Next, define the routes by providing additional properties to the MaterialApp constructor:the initialRoute and the routes themselves.

The initialRoute property defines which route the app should start with. The routes propertydefines the available named routes and the widgets to build when navigating to those routes.

Navigate to the second screen

With the widgets and routes in place, trigger navigation by using the Navigator.pushNamed() method. This tells Flutter to build the widget defined inthe routes table and launch the screen.

In the build() method of the FirstScreen widget, update the onPressed()

callback:Return to the first screen

To navigate back to the first screen, use the Navigator.pop() function.

NavigationRail is a Flutter widget that allows creating a navigation bar at the left or right location of the screen. Like the bottom navigation bar which sticks to the bottom, we have a navigation rail that stays on one of the sides. NavigationRail is suitable for large viewports likeon desktop or tablet.

Parameters:

1. backgroundColor: The color of the container which holds the navigation rail.
2. destinations: The destinations represented by items on the rails to be taken.
3. elevation: The navigation rail's elevation.
4. extended: Boolean to set if rail is to be extended.
5. groupAlignment: The vertical alignment of the items of rails.
6. indicatorColor: The color of indicated items.
7. labelType: The styling of the text of items on rails.
8. leading: The widget that appears before destinations.
9. onDestinationSelected: Function to be called when any destination is selected.
10. minExtendedWidth: The width to be extended after the rail layout on the screen.
11. minWidth: The smallest possible width of rails regardless of destinations icon.
12. selectedIconTheme: The style of icon selected.
13. selectedIndex: The index of the item that is selected from the destinations array.
14. selectedLabelTextStyle: The text style of icon label that is selected from destinations.
15. trailing: The widget that is displayed below the destinations.

16. unselectedIconTheme: The styling of unselected items among the destinations.
17. unselectedLabelTextStyle: The text style of labels of destinations.
18. useIndicator: Boolean when set to true creates a box behind a selected icon ofdestinations on rails.

Code:

Routing

```
import 'package:flutter/material.dart';
import "dart:math";

class homePage extends StatelessWidget {
  const homePage();
  @override
  Widget build(BuildContext context) {
    return Scaffold(
        body: Center(
      child: ElevatedButton(
        child: Text("click on the button to see next page"),
        onPressed: () {
          Navigator.of(context).pushNamed(RouteGenerator.randompage);
        },
      ),
    ));
  }
}

void main() {
  runApp(const RandomApp());
}

class RandomApp extends StatelessWidget {
  const RandomApp();
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      onGenerateTitle: (_) => "Random App",
      initialRoute: RouteGenerator.homepage,
      onGenerateRoute: RouteGenerator.generateRoute,
    );
  }
}

class randomPage extends StatelessWidget {
  const randomPage();
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Center(
        child: Text("${Random().nextInt(50)}"),
      ),
    );
  }
}
```

```
class RouteGenerator {
  static const String homepage = '/';
  static const String randompage = '/random';
  RouteGenerator._() {}

  static Route<dynamic> generateRoute(RouteSettings settings) {
    switch (settings.name) {
      case homepage:
        {
          return MaterialPageRoute(builder: (_) => const homePage());
        }
      case randompage:
        {
          return MaterialPageRoute(builder: (_) => const randomPage());
        }
      default:
        {
          throw FormatException("Route not found");
        }
    }
  }
}

class RouteException implements Exception {
  final String message;
  const RouteException(this.message);
}
```

Passing Data using Navigator

```
import 'package:flutter/material.dart';

void main() {
  runApp(const Myapp());
}

class Details_of_List extends StatelessWidget {
  final Todo item;
  const Details_of_List(this.item);

  @override
  Widget build(BuildContext context) {
    return Scaffold(
        appBar: AppBar(title: Text("Aakash Dhotre")),
        body: Center(
            child: Text(
          "${item.title}\n${item.description}",
          textScaleFactor: 4,
        )));
    ;
  }
}
```

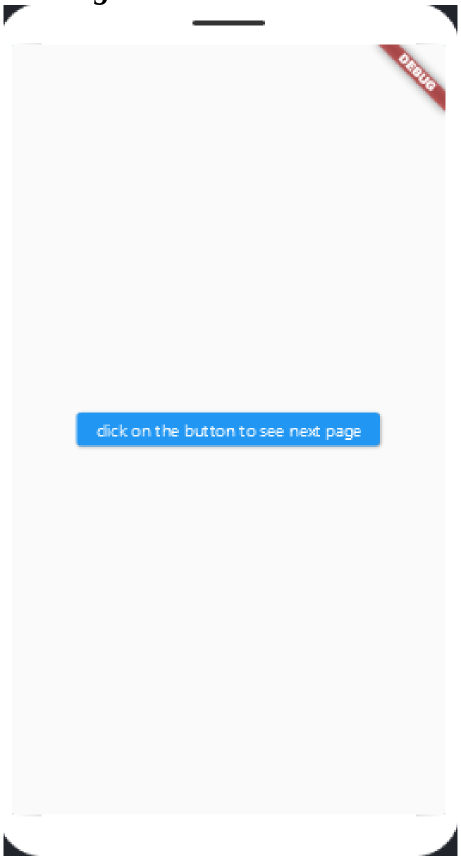
```
class HomePage extends StatelessWidget {
  HomePage();
  final List<Todo> todolist = [
    const Todo("title 1", "this is the title 1"),
    const Todo("title 2", "this is the title 2"),
    const Todo("title 3", "this is the title 3"),
    const Todo("title 4", "this is the title 4")
  ];
  void _itemPressed(BuildContext context, Todo item) {
    Navigator.of(context)
        .push(MaterialPageRoute(builder: (_) => Details_of_List(item)));
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
        appBar: AppBar(title: Text("Aakash Dhotre")),
        body: ListView.builder(
          itemCount: todolist.length,
          itemBuilder: (context, value) {
            return ListTile(
              title: Text(todolist[value].title, textScaleFactor: 2),
              subtitle: Text(todolist[value].description, textScaleFactor: 2),
              onTap: () {
                _itemPressed(context,
                    Todo(todolist[value].title, todolist[value].description));
              },
            );
          },
        ));
  }
}

class Myapp extends StatelessWidget {
  const Myapp();

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      onGenerateTitle: (_) => 'Passing data with Navigator',
      initialRoute: RouteGenerator.homepage,
      onGenerateRoute: RouteGenerator.generateRoute,
    );
  }
}

class Todo {
  final String title;
  final String description;
  const Todo(this.title, this.description);
}

class RouteGenerator {
  static const String homepage = '/';
  static const String details_of_list = '/details_of_list';
  RouteGenerator._() {}
  static Route<dynamic> generateRoute(RouteSettings settings) {
    switch (settings.name) {
      case homepage:
        {
          return MaterialPageRoute(builder: (_) => HomePage());
        }

      default:
        {
          throw FormatException("Route not found");
        }
    }
  }
}

class RouteException implements Exception {
  final String message;
  const RouteException(this.message);
}
```

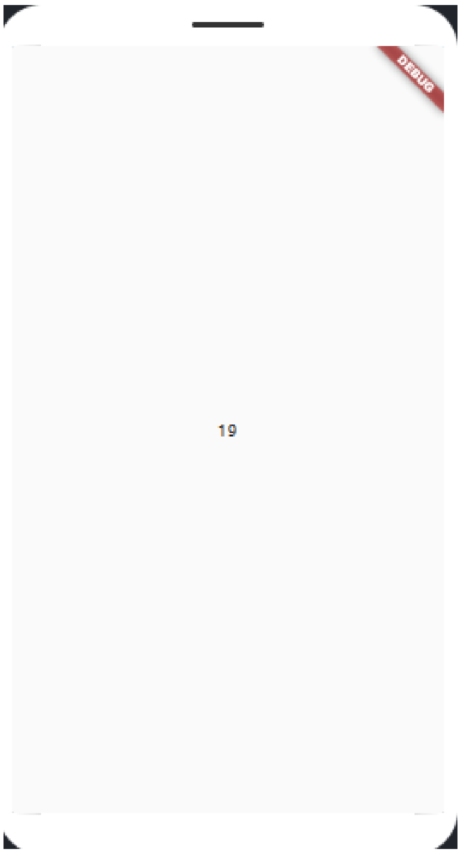
Navigation Rail

```
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';

void main() {
  runApp(ChangeNotifierProvider<IndexProvider>(
      create: (_) => IndexProvider(), child: MyApp()));
}

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
        home: Scaffold(
            appBar: AppBar(title: Text("BottomNavigation Bar")),
            bottomNavigationBar: Buttomnavigation()));
  }
}

class Buttomnavigation extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Consumer<IndexProvider>(builder: (context, IP, _) {
      return BottomNavigationBar(
        currentIndex: IP.index,
        selectedItemColor: Colors.green,
        onTap: (int itemnumber) {
          IP.set_index(itemnumber);
        },
        items: [
          BottomNavigationBarItem(
              icon: Icon(
                Icons.home,
                color: Colors.red,
              ),
              label: "Home"),
          BottomNavigationBarItem(
              icon: Icon(Icons.mail, color: Colors.red), label: "Mail"),
          BottomNavigationBarItem(
              icon: Icon(Icons.settings, color: Colors.red), label: "Setting"),
          BottomNavigationBarItem(
              icon: Icon(Icons.contacts, color: Colors.red),
              label: "Contact Us"),
        ],
      );
    });
  }
}

class IndexProvider with ChangeNotifier {
  int index = 1;
  void set_index(int value) {
    index = value;
    notifyListeners();
    debugPrint("value is updated $value");
  }

  int get_index() {
    return index;
  }
}
```

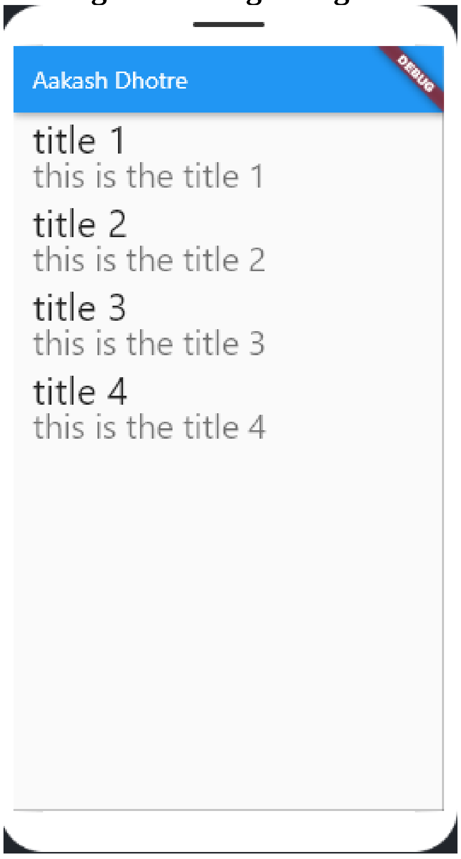
Output:

Routing

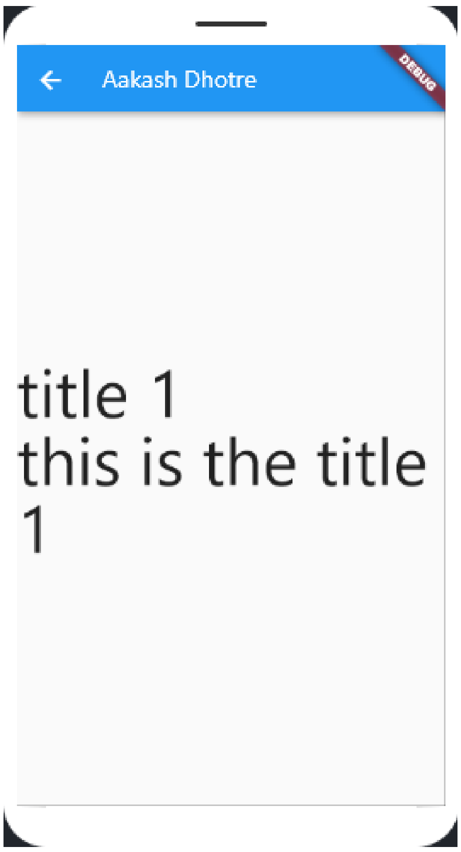




Passing Data using Navigator

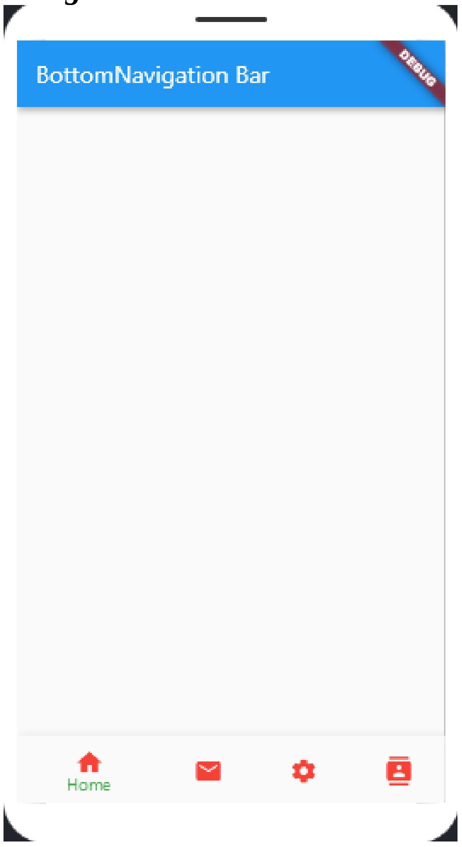


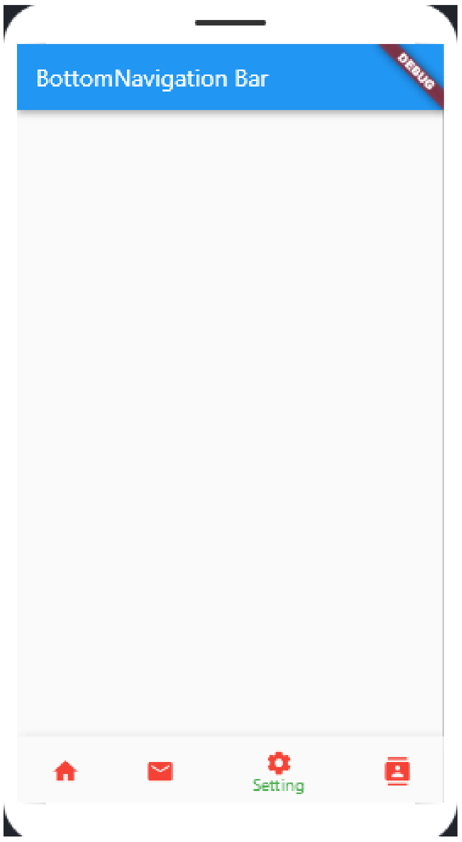
title 1
this is the title 1
title 2
this is the title 2
title 3
this is the title 3
title 4
this is the title 4



title 1
this is the title 1

Navigation Rail





Conclusion:

Application (write two Application)

Questionnaire:

1. What is the future of Flutter app development?

2. Explain the types of streams in Dart

3. What are some approaches to state management?

4. Explain state management

5. What is Widget Testing?

6. What is the BuildContext?

7. What are the available build modes?

8. How do you use keys?

9. How do you add interactivity to an app?

10. What are some drawbacks of Flutter and Dart?

Experiment No 07

Aim: create a simple hello world PWA

Theory:

Progressive Web Apps the basic idea is to use browser technologies to build a web application that works offline and has the look and feel of a native application. In this tutorial I'll show you how to use a manifest and service workers to create just about the simplest app possible, one thatworks without an internet connection, and can be added to your home screen.

To get the most out of this tutorial you should be familiar with HTML, CSS and JavaScript. If you can code a web page and use plain-vanilla JavaScript to add some interactivity you should be able to follow along. To build this app you'll need a text editor, the latest version of Chromeand a local web server. I've used Adobe's Brackets in this tutorial, because it has a built in webserver, but you can use any text editor and server combo you're comfortable with.

Create a directory for the app and add *js*, *css*, and *images* subdirectories. It should look like thiswhen you're finished:

```
/Hello-PWA # Project Folder
/css # Stylesheets
/js # Javascript
/images # Image files.
```

Writing the App Interface

When writing the markup for a Progressive Web App there are 2 requirements to keep in mind:

1. The app should display some content even if JavaScript is disabled. This prevents users from seeing a blank page if their internet connection is bad or if they are using an olderbrowser.
2. It should be responsive and display properly on a variety of devices. In other words, itneeds to be mobile friendly.
   For our little app, we'll tackle the first requirement by simply hard coding the content andthe second by adding a viewport meta tag.
   To do this, create a file named *index.html* in your project root folder

Add a Manifest

The final requirement for a PWA is to have a manifest file. The manifest is a *json* file that is usedto specify how the app will look and behave on devices. For example, you can set the app's orientation and theme color.
Save a file named *manifest.json* in your root folder and add the following content:Line-by-line the fields are as follows:

name
The title of the app. This is used when prompting the user to install the app. It should be the fulltitle of the app.
short_name
Is the name off the app as it will appear on the app icon. This should be short and to the point.
lang
The default language the app is localized in. In our case, English.
start_url
Tells the browser which page to load up when the app is launched. This will usuallybe *index.html* but it doesn't need to be.
display
The type of shell the app should appear in. For our app, we are using *standalone* to make it lookand feel like a standard native app. There are other settings to make it full screen or include the browser chrome.
background_color
The color of the splash screen that opens when the app launches.
theme_color
Sets the color of the tool bar and in the task switcher.

Code:

1. To do this, create a file named index.html in your project root folder and add the following markup:

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Hello World</title>
  <link rel="stylesheet" href="css/style.css">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
<body class="fullscreen">
  <div class="container">
    <h1 class="title">Hello World!</h1>
  </div>
</body>
</html>
```

2. Next, create a file named style.css in the css folder and add this code:

```
body {
  font-family: sans-serif;
}

/* Make content area fill the entire browser window */
html,
.fullscreen {
  display: flex;
  height: 100%;
  margin: 0;
  padding: 0;
  width: 100%;
}

/* Center the content in the browser window */
.container {
  margin: auto;
  text-align: center;
}

.title {
  font-size: 3rem;
}
```

3. You can now test the app by clicking on the preview button in Brackets. (The lightning bolt in the upper right-hand corner.) This will open a Chrome window and serve up your page.

4. Press F12 to open the developer panel in Chrome and click on the audits tab to open Lighthouse.

5. Create a file named sw.js in your root folder and enter the contents of the script below.

```
var cacheName = 'hello-pwa';
var filesToCache = [
  '/',
  '/index.html',
  '/css/style.css',
  '/js/main.js'
];

/* Start the service worker and cache all of the app's content */
self.addEventListener('install', function(e) {
  e.waitUntil(
    caches.open(cacheName).then(function(cache) {
      return cache.addAll(filesToCache);
    })
  );
});

/* Serve cached content when offline */
self.addEventListener('fetch', function(e) {
  e.respondWith(
    caches.match(e.request).then(function(response) {
      return response || fetch(e.request);
    })
  );
});
```

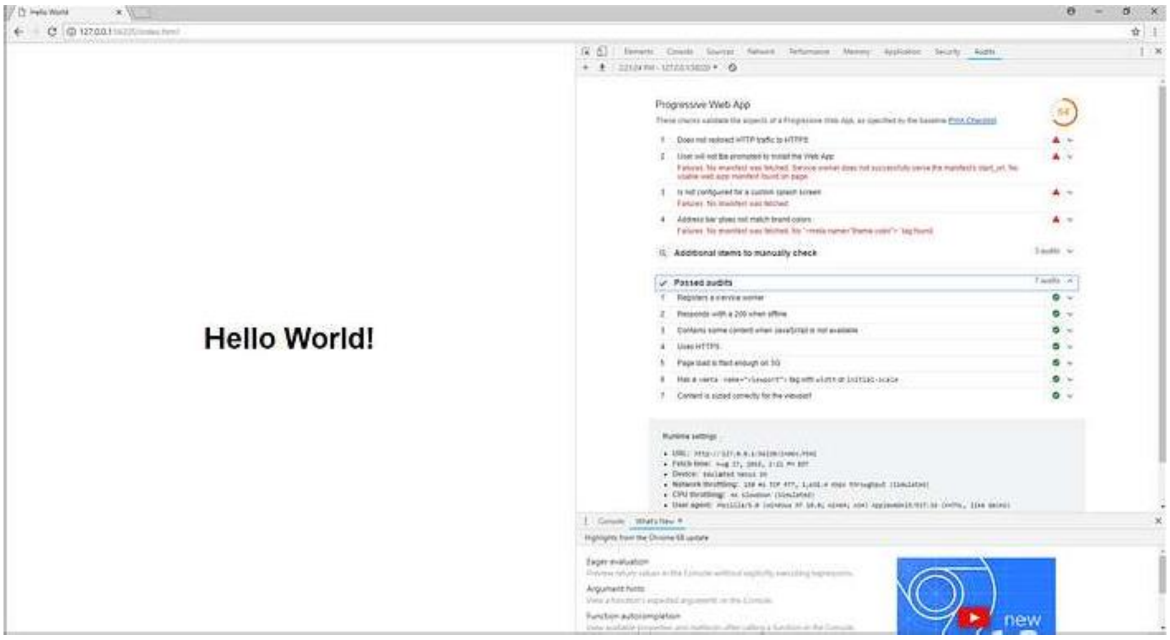
6. Create a file named main.js in the js folder and enter the following code:

```
window.onload = () => {
  'use strict';

  if ('serviceWorker' in navigator) {
    navigator.serviceWorker
             .register('./sw.js');
  }
}
```

7. The revised index.html should look like this:

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Hello World</title>
  <link rel="stylesheet" href="css/style.css">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
</head>
<body class="fullscreen">
  <div class="container">
    <h1 class="title">Hello World!</h1>
  </div>
  <script src="js/main.js"></script>
</body>
</html>
```

Output:



Hello World!

Progressive Web App

These checks validate the aspects of a Progressive Web App, as specified by the baseline PWA Checklist

1 Does not redirect HTTP traffic to HTTPS
2 User will not be prompted to install the Web App
Failures: No manifest was fetched. Service worker does not successfully serve the manifest's start_url. No viable web app manifest found on page.
3 Is not configured for a custom splash screen
Failures: No manifest was fetched.
4 Address bar does not match brand colors
Failures: No manifest was fetched. No `<meta name="theme-color">` tag found.

Additional items to manually check

Passed audits
1 Registers a service worker
2 Responds with a 200 when offline
3 Contains some content when JavaScript is not available
4 Uses HTTPS
5 Page load is fast enough on 3G
6 Has a `<meta name="viewport">` tag with `width` or `initial-scale`
7 Content is sized correctly for the viewport

Runtime settings
- URL: [illegible]
- Fetch time: [illegible]
- Device: [illegible]
- Network throttling: [illegible]
- CPU throttling: [illegible]
- User agent: [illegible]

Conclusion:

Application (write two Application)

Questionnaire:

1. What is a progressive web app?

2. Do all browsers support PWA?

3. What are business benefits of PWA implementation?

4. What makes an app a PWA?

5. How do Progressive Web Apps work?

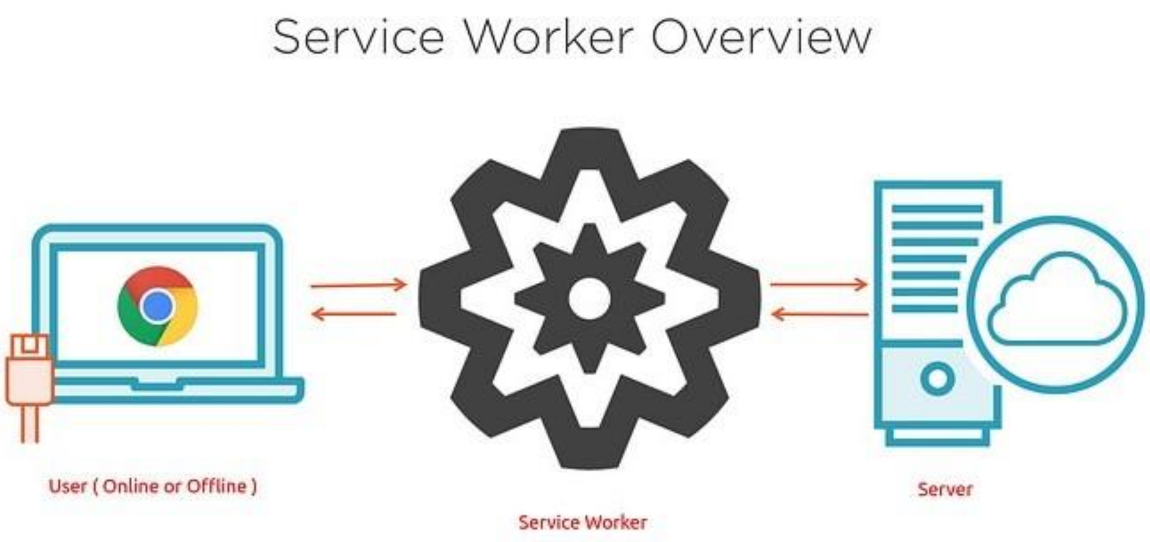
Experiment No 08

Aim: To code and register a service worker, and complete the install and activation process for anew service worker for the E-commerce PWA.

Theory:

Service Worker

A service worker is an event-driven worker registered against an origin and a path.
It is simply a JavaScript file that can control the web page by intercepting and modifying navigation and resource requests and caching resources such that there is complete control overhow to make the app behave in certain situations such as for offline experience.
Service Workers act as proxy servers that sit between web applications, the browser, and thenetwork. They are intended to enable the creation of effective offline experiences through intercepting network requests.



.

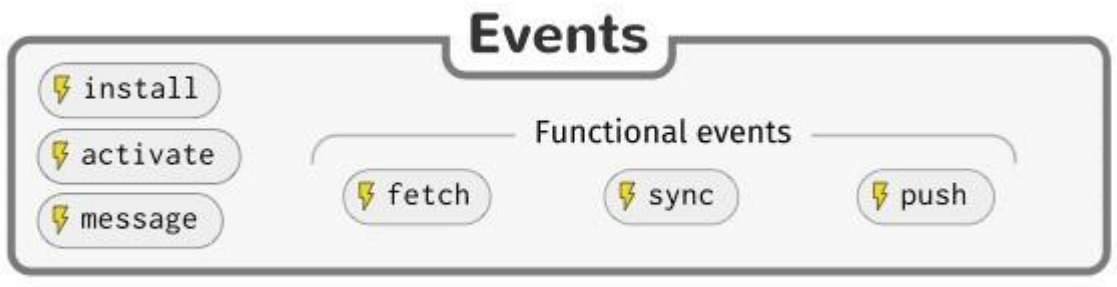
<u>Basic architecture</u>

With service workers, the following steps are generally observed for basic setup:

1. The service worker URL is fetched and registered via serviceWorkerContainer.register().
2. If successful, the service worker is executed in a ServiceWorkerGlobalScope, which is aspecial kind of worker context, running off the main script execution thread, with no DOM access.
3. The service worker is now ready to process events.
4. Installation of the worker is attempted when service worker-controlled pages are accessed. An Install event is always the first one sent to a service worker (This can beused to start the process of caching site assets).

5. When the oninstall handler completes, the service worker is considered installed.
6. Next is activation. When the service worker is installed, it then receives an activate event.The primary use of onactivateis for cleanup of resources used in previous versions of a Service worker script.
7. The Service worker will now control pages that are loaded after the register() issuccessful. So documents will have to be reloaded to actually be controlled.

The below graphic shows a summary of the available service workerevents:



Registration

A service worker is first registered using the ServiceWorkerContainer.register() method. If successful, the service worker will be downloaded to the client which will attempt installation/activation for URLs accessed by the user inside the whole origin, or inside a subsetspecified through the scope.

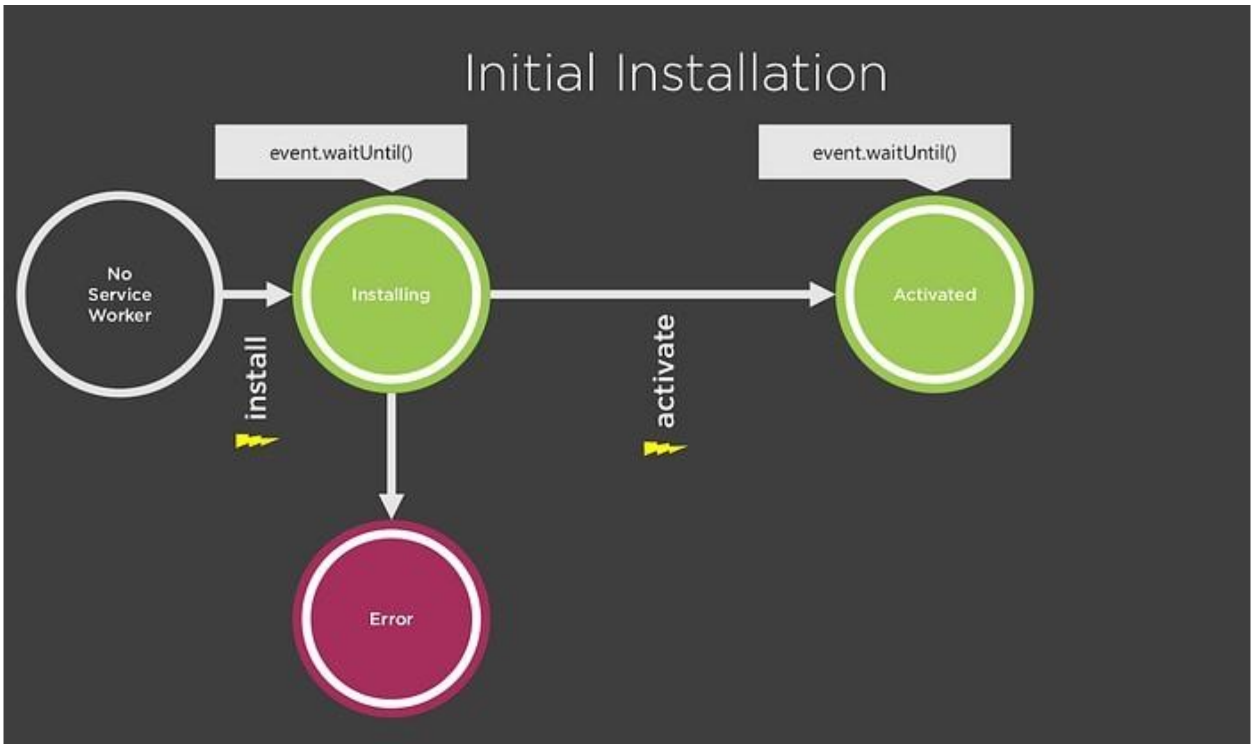
```
main.js// Registers the service worker if supported by browserif
(navigator.serviceWorker) {
 navigator.serviceWorker
   .register('sw.js', { scope: '/' })
   .then(() => console.log('Service Worker: Registered'))
   .catch(() => console.log('Service Worker Registration: Error'));
}
```

Lifecycle of Service Worker

At this point, the service worker will observe the following lifecycle:

Download
Install
Activate

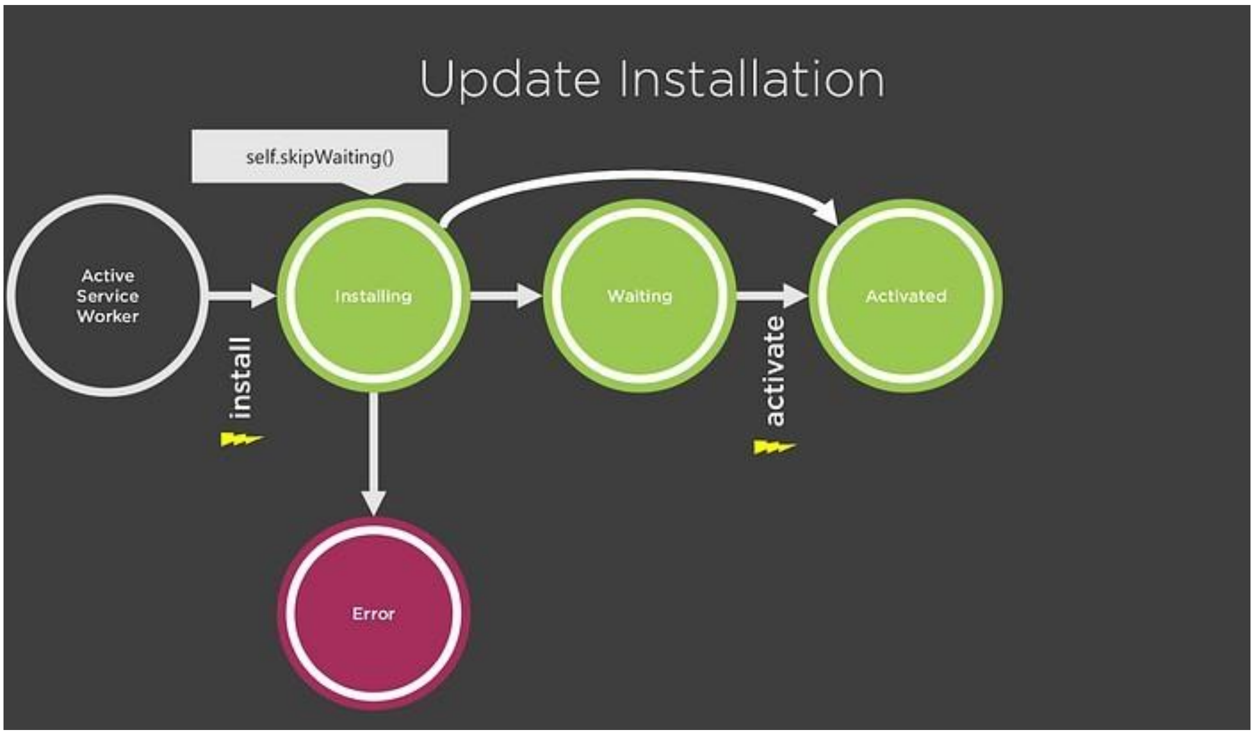
a. Initial Installation



```
sw.js...self.addEventListener('install', (event) => {
 console.log('Service Worker: Installed');
...// Store cache for offline experience...
});...
```

Update Installation

If there is an existing service worker available, the new version is installed in the background,but not yet activated — at this point it is called the worker in waiting.

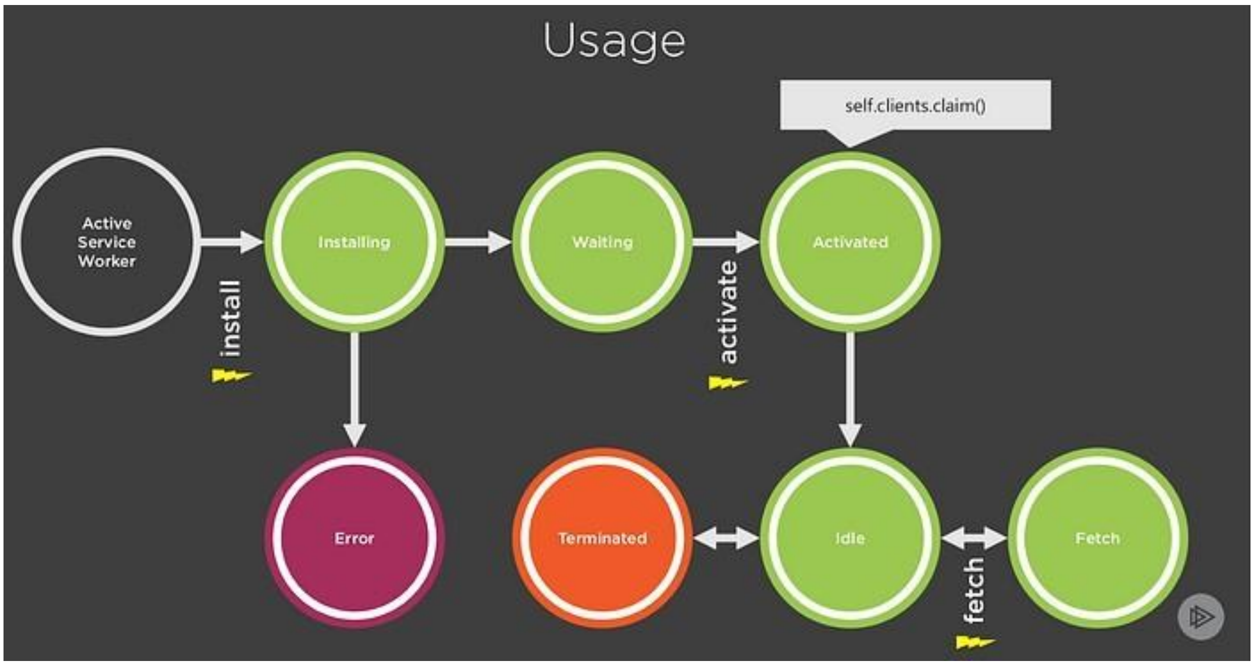


```
sw.js// Manually skipWaiting the waiting state via
self.skipWaiting()...self.addEventListener('install', (e) => {
 console.log('Service Worker: Installed');
 self.skipWaiting();
});
```

Activation

```
const cacheName = 'v2';…

self.addEventListener('activate', e => {
   console.log('Service Worker: Activated');e.waitUntil(
       caches.keys().then(cacheNames => {
           return Promise.all(
               cache Names.map(cache => {
                   if (cache !== cacheName) {
                       console.log('Service Worker: Clearing Old
Cache');
```

```
                                return caches.delete(cache);
                           }
                      })
                 );
            })
       );
});
```



Code:
index.html

```
<!DOCTYPE html>
An image will appear here in 3 seconds:
<script>
  navigator.serviceWorker.register('/sw.js')
    .then(reg => console.log('SW registered!', reg))
    .catch(err => console.log('Boo!', err));

  setTimeout(() => {
    const img = new Image();
    img.src = '/dog.svg';
    document.body.appendChild(img);
  }, 3000);
</script>
```

sw.js

```
self.addEventListener('install', event => {
  console.log('V1 installing…');

  // cache a cat SVG
  event.waitUntil(
    caches.open('static-v1').then(cache => cache.add('/cat.svg'))
  );
});

self.addEventListener('activate', event => {
  console.log('V1 now ready to handle fetches!');
});

self.addEventListener('fetch', event => {
  const url = new URL(event.request.url);

  // serve the cat SVG from the cache if the request is
  // same-origin and the path is '/dog.svg'
  if (url.origin == location.origin && url.pathname == '/dog.svg') {
    event.respondWith(caches.match('/cat.svg'));
  }
});
```

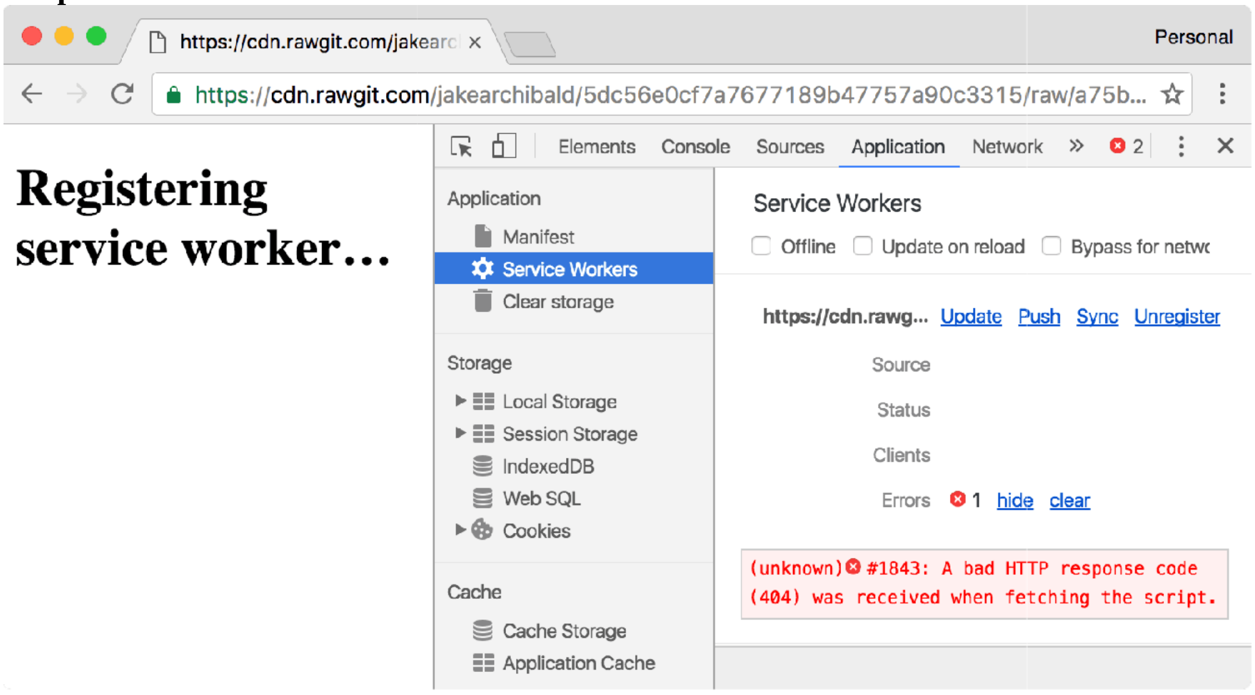
Let's say we changed our service worker script to respond with a picture of a horse rather than a cat:

```
const expectedCaches = ['static-v2'];

self.addEventListener('install', event => {
  console.log('V2 installing…');

  // cache a horse SVG into a new cache, static-v2
  event.waitUntil(
    caches.open('static-v2').then(cache => cache.add('/horse.svg'))
  );
});

self.addEventListener('activate', event => {
  // delete any caches that aren't in expectedCaches
  // which will get rid of static-v1
  event.waitUntil(
    caches.keys().then(keys => Promise.all(
      keys.map(key => {
        if (!expectedCaches.includes(key)) {
          return caches.delete(key);
        }
      })
    )).then(() => {
      console.log('V2 now ready to handle fetches!');
    })
  );
});

self.addEventListener('fetch', event => {
  const url = new URL(event.request.url);
  // serve the horse SVG from the cache if the request is
  // same-origin and the path is '/dog.svg'
  if (url.origin == location.origin && url.pathname == '/dog.svg') {
    event.respondWith(caches.match('/horse.svg'));
  }
});
```

Handling updates

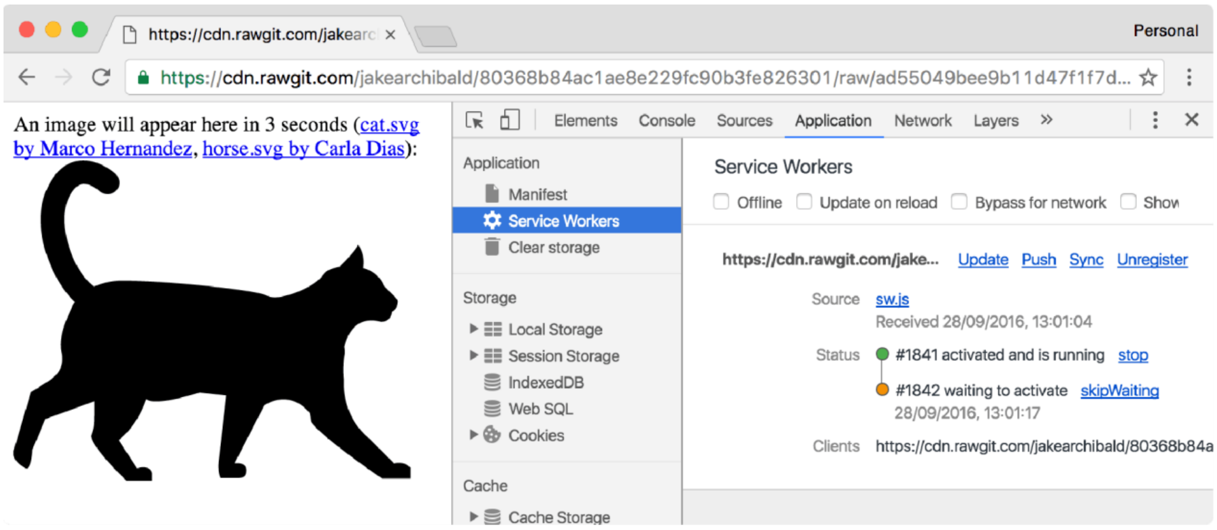
```
navigator.serviceWorker.register('/sw.js').then(reg => {
  reg.installing; // the installing worker, or undefined
  reg.waiting; // the waiting worker, or undefined
  reg.active; // the active worker, or undefined

  reg.addEventListener('updatefound', () => {
    // A wild service worker has appeared in reg.installing!
    const newWorker = reg.installing;

    newWorker.state;
    // "installing" - the install event has fired, but not yet complete
    // "installed"  - install complete
    // "activating" - the activate event has fired, but not yet complete
    // "activated"  - fully active
    // "redundant"  - discarded. Either failed install, or it's been
    //                replaced by a newer version

    newWorker.addEventListener('statechange', () => {
      // newWorker.state has changed
    });
  });
});

navigator.serviceWorker.addEventListener('controllerchange', () => {
  // This fires when the service worker controlling this page
  // changes, eg a new worker has skipped waiting and become
  // the new active worker.
});
```

Output:



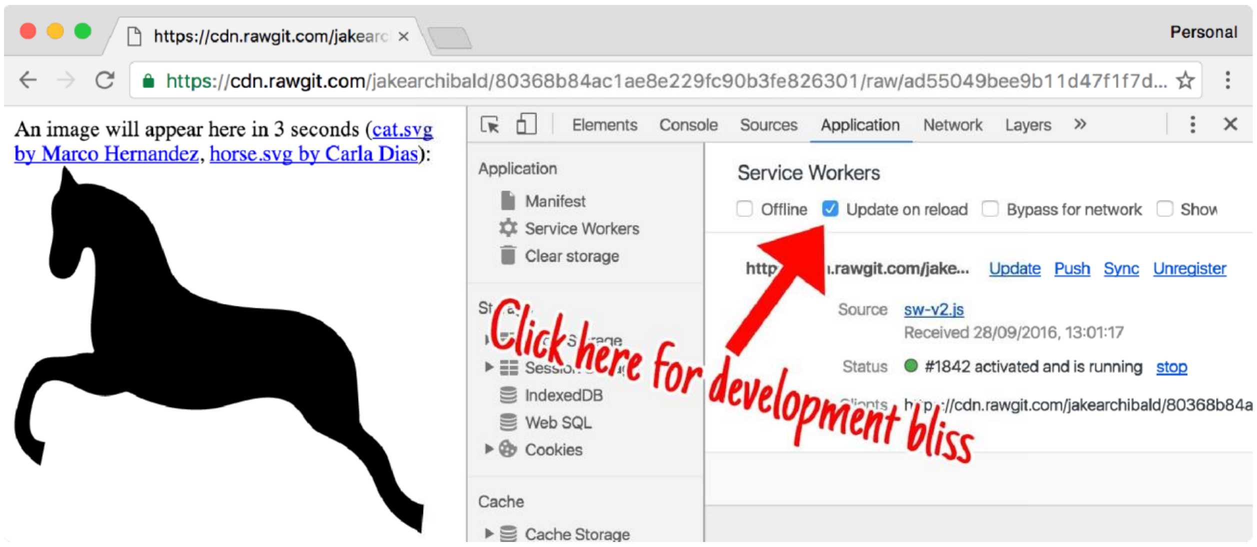
Registering service worker…

```
(unknown) #1843: A bad HTTP response code
(404) was received when fetching the script.
```



An image will appear here in 3 seconds (cat.svg by Marco Hernandez, horse.svg by Carla Dias):

Source sw.js
Received 28/09/2016, 13:01:04
Status #1841 activated and is running stop
#1842 waiting to activate skipWaiting
28/09/2016, 13:01:17
Clients https://cdn.rawgit.com/jakearchibald/80368b84a



https://cdn.rawgit.com/jakearc ×

Personal

https://cdn.rawgit.com/jakearchibald/80368b84ac1ae8e229fc90b3fe826301/raw/ad55049bee9b11d47f1f7d...

An image will appear here in 3 seconds (cat.svg by Marco Hernandez, horse.svg by Carla Dias):

Elements Console Sources Application Network Layers »

Application: Manifest, Service Workers, Clear storage

Storage: Local Storage, Session Storage, IndexedDB, Web SQL, Cookies

Cache: Cache Storage

Service Workers

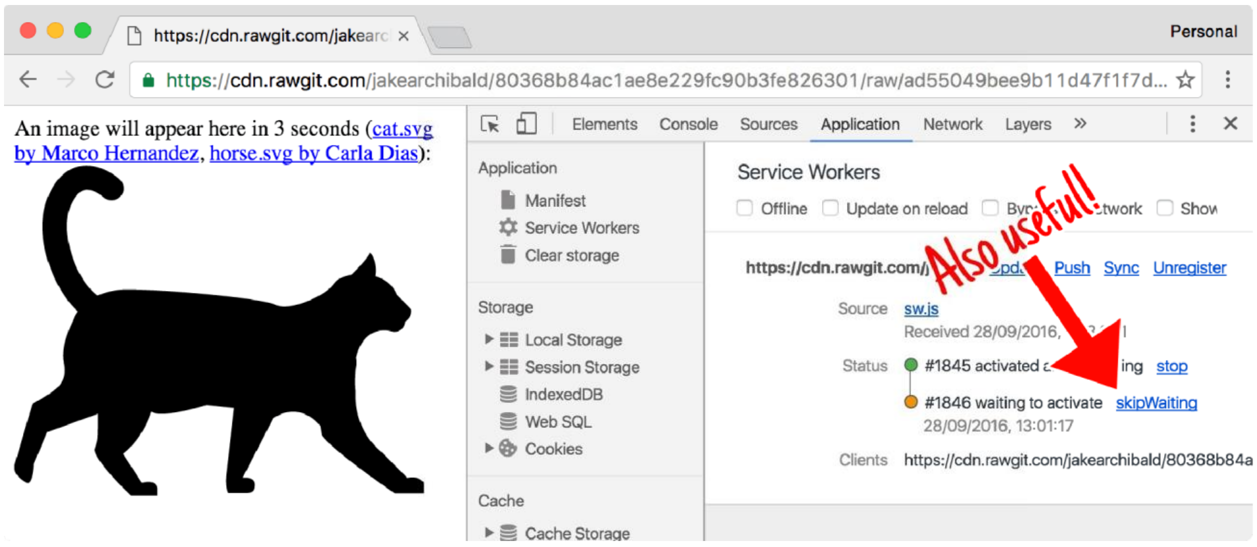
☐ Offline ☑ Update on reload ☐ Bypass for network ☐ Show

http...rawgit.com/jake... Update Push Sync Unregister

Source sw-v2.js
Received 28/09/2016, 13:01:17

Status ● #1842 activated and is running stop

Clients http://cdn.rawgit.com/jakearchibald/80368b84a



https://cdn.rawgit.com/jakearc ×

Personal

https://cdn.rawgit.com/jakearchibald/80368b84ac1ae8e229fc90b3fe826301/raw/ad55049bee9b11d47f1f7d...

An image will appear here in 3 seconds (cat.svg by Marco Hernandez, horse.svg by Carla Dias):

Elements Console Sources Application Network Layers »

Application: Manifest, Service Workers, Clear storage

Storage: Local Storage, Session Storage, IndexedDB, Web SQL, Cookies

Cache: Cache Storage

Service Workers

☐ Offline ☐ Update on reload ☐ Bypass for network ☐ Show

https://cdn.rawgit.com/... Update Push Sync Unregister

Source sw.js
Received 28/09/2016, ...

Status ● #1845 activated and is running stop
● #1846 waiting to activate skipWaiting
28/09/2016, 13:01:17

Clients https://cdn.rawgit.com/jakearchibald/80368b84a

Conclusion:

Application (write two Application)

Questionnaire:

1. What are some requirements to make the website installable as PWA?

2. What features do Progressive Web Apps have that native apps lacks?

3. What is CacheStorage?

4.What is IndexedDB and how is it used by PWA?

5. What is a fetch event?

Experiment No 09

Aim: To implement Service worker events like fetch, sync and push for E-commerce PWA.

Theory:

Fetch Event

```
self.addEventListener('fetch',
      function(event) {event.respondWith(
        caches.match(event.request)
          .then(function(respons
            e) {if (response) {
              return response;
            }
            return  fetch(event.request);
          }
        )
      );
    });
```

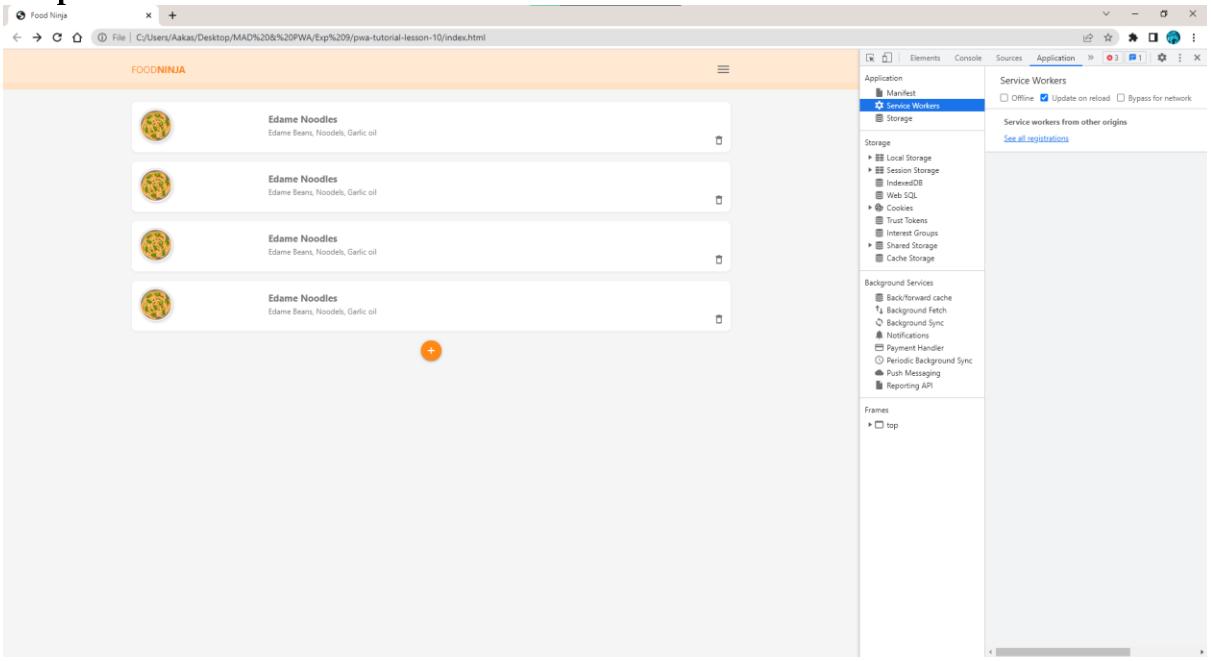
Background Sync

```
// Register your service worker:
    navigator.serviceWorker.register('/sw.js');

    // Then later, request a one-off sync:
    navigator.serviceWorker.ready.then(function(swRegistration) {
      return   swRegistration.sync.register('myFirstSync');
    });


self.addEventListener('sync',
      function(event)  {if (event.tag ==
      'myFirstSync') {
        event.waitUntil(doSomeStuff());
  }
});
```

Push Event

```
self.addEventListener(
"push",
(event) => {
let message = event.data.json();
switch (message.type) {
case "init":
doInit();
break;
case "shutdown":
doShutdown();
break;
}
},
false
);
```

Output:



Conclusion:

Application (write two Application)

Questionnaire:

1. What is a service worker?

2. What is the differences between a Hybrid Mobile App and a Progressive Web App?

3. How to update a service worker?

4. What are some requirements to app shell?

5. Is it possible to have multiple service workers?

Experiment No 10

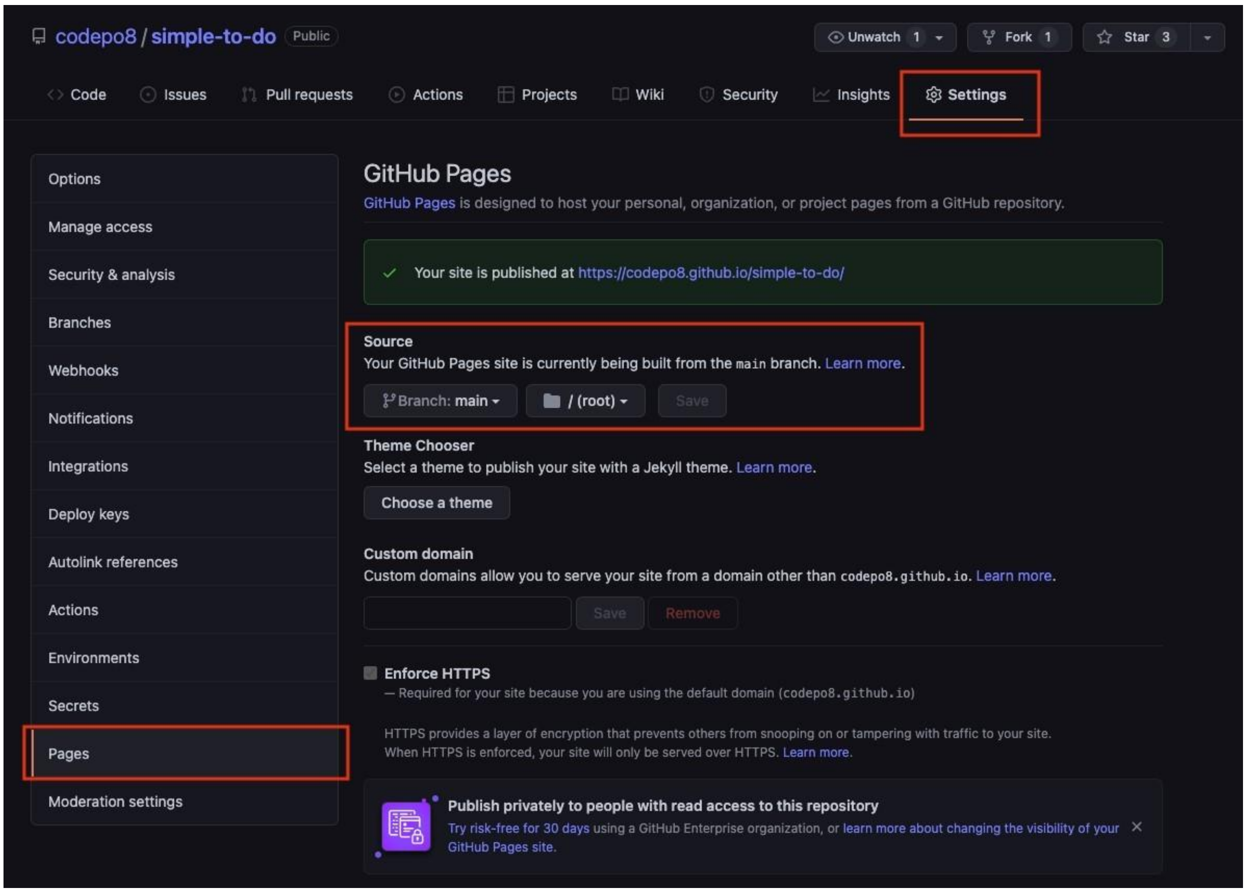
Aim: To implement Service worker events like fetch, sync and push for E-commerce PWA.

Theory:

This is a bare-bones example on how to turn an index.html document on GitHub and hosted as aGitHub Page into an installable Progressive Web App with offline caching.

As a reminder, you can host HTML, CSS and JavaScript files on GitHub as pages. For example,I have bare bones To Do app at https://github.com/codepo8/simple-to-do with an index.html document.

In the settings of this repository simple-to-do, I chose to publish the main branch as a GitHubpage as shown in the following screenshot.



codepo8 / simple-to-do Public

GitHub Pages

GitHub Pages is designed to host your personal, organization, or project pages from a GitHub repository.

Your site is published at https://codepo8.github.io/simple-to-do/

Source

Your GitHub Pages site is currently being built from the main branch. Learn more.

Branch: main / (root) Save

Theme Chooser

Select a theme to publish your site with a Jekyll theme. Learn more.

Choose a theme

Custom domain

Custom domains allow you to serve your site from a domain other than codepo8.github.io. Learn more.

Enforce HTTPS

— Required for your site because you are using the default domain (codepo8.github.io)

HTTPS provides a layer of encryption that prevents others from snooping on or tampering with traffic to your site. When HTTPS is enforced, your site will only be served over HTTPS. Learn more.

Publish privately to people with read access to this repository

Try risk-free for 30 days using a GitHub Enterprise organization, or learn more about changing the visibility of your GitHub Pages site.

This means that this app is now available at https://codepo8.github.io/simple-to-do/. Every time Ipublish to the main branch, it triggers an action and the page is generated.

This here is a template repository that does not only publish the page, but also offers it as aninstallable app and shows the page when the user is offline.

Changing the index.html

The first thing you need to do change is the index.html document. You need two things for this.Your GitHub username, for example in this case codepo8 and the name of the repository you host as a GitHub Page, in this case github-page-pwa.

The current index.html has these settings already, and you need to change them accordingly.

In the following example, each codepo8 needs to become yours and github-page-pwa the nameof your repository. Make sure to not remove any /, as they are crucial for this to work.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>GitHub page as PWA template</title>
  <link rel="canonical" href="https://codepo8.github.io/github-page-pwa/" />
  <link rel="manifest" href="/github-page-pwa/manifest.webmanifest">
</head>
<body>
  <h1>GitHub page as PWA template</h1>
    …
  <script>
      if (navigator.serviceWorker) {
        navigator.serviceWorker.regi
        ster (
          '/github-page-pwa/sw.js',
          {scope: '/github-page-pwa/'}
        )
      }
  </script>
</body>
</html>
```

Changing the service worker to make your site available offline

The sw.js file is the ServiceWorker that defines which of the files in your application shouldbecome available offline. Again you need to change some settings to your needs.

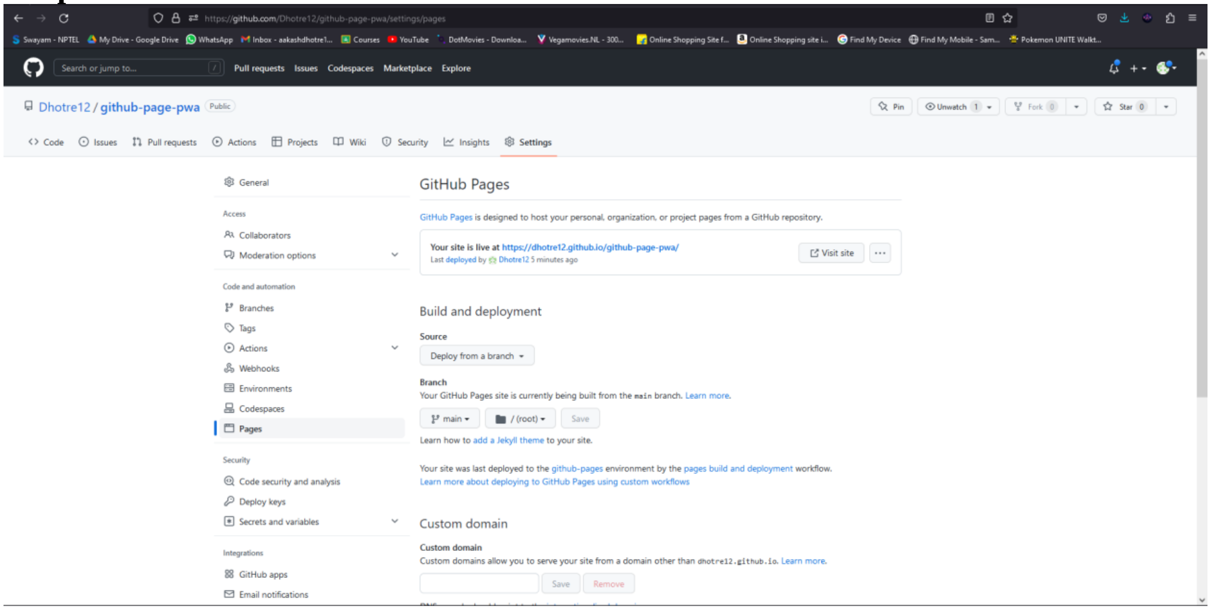
```
// Change this to your
repository namevar GHPATH =
'/github-page-pwa';

// Choose a different app
prefix namevar APP_PREFIX =
'gppwa_';

// The version of the cache. Every time you change any of the files
// you need to change this version (version_01, version_02…).
// If you don't change the version, the service worker will give your
// users the old files!
var VERSION = 'version_00';

// The files to make available for offline use. make sure to add
// others to
this listvar
URLS = [
  `${GHPATH}/`,
  `${GHPATH}/index.html`,
  `${GHPATH}/css/styles.css`,
  `${GHPATH}/js/app.js`
]
```

Changing the manifest to make your app installable

The manifest.webmanifest file defines the name and look of the GitHub Page as an installableapplication. You need to change the names, description, URLs and link to the icon of the application to your needs. I added comments here as to what is what.

```
{
  // Name of the app and short name in case there isn't enough
  space"name": "Github Page PWA",
  "short_name": "GPPWA",
  // Description what your app is
  "description": "Github Page as a Progressive Web App",

  // Scope and start URL - these need to change to yours
  "scope": "/github-page-pwa/",
  "start_url": "/github-page-pwa/",

  // colours of the app as displayed in the
  installer"background_color": "#ffffff",
  "theme_color": "#ffffff",

  // Display of the app.
  //This could be "standalone", "fullscreen", "minimal-ui" or
  "browser""display": "standalone",

  // The possible icons to display. Make sure to change the src URL,
  // the type and the size to your needs. If the size isn't correct,
  // you may not be able to install
  the app."icons": [
      {
        "src": "/github-page-
        pwa/img/icon.png","type":
        "image/png",
        "sizes": "700x700"
      }
  ]
}
```

Output:



https://github.com/Dhotre12/github-page-pwa/settings/pages

Dhotre12 / github-page-pwa Public

GitHub Pages

GitHub Pages is designed to host your personal, organization, or project pages from a GitHub repository.

Your site is live at https://dhotre12.github.io/github-page-pwa/
Last deployed by Dhotre12 5 minutes ago

Build and deployment

Source
Deploy from a branch

Branch
Your GitHub Pages site is currently being built from the `main` branch. Learn more.

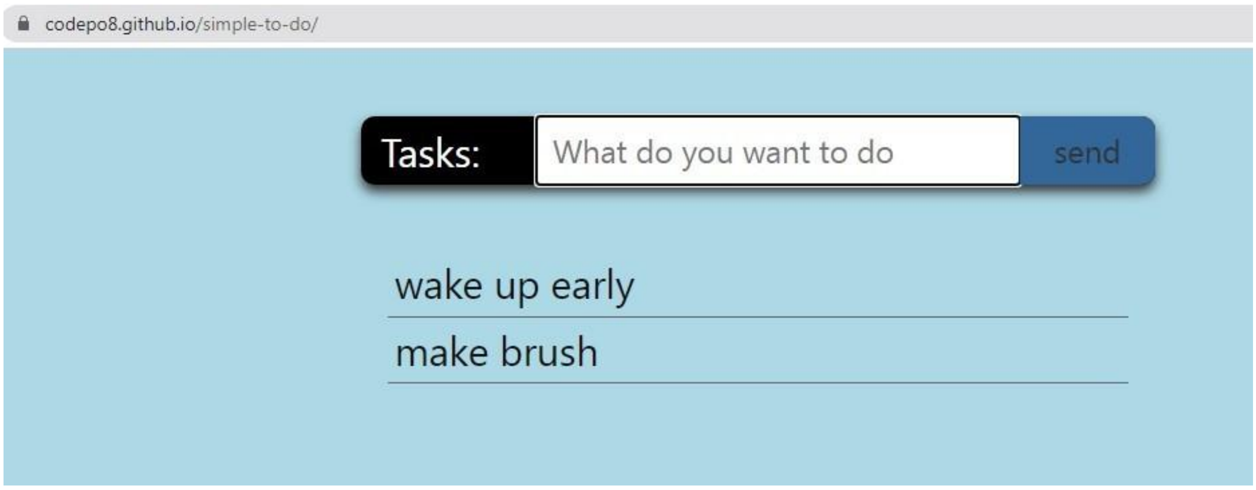
main / (root) Save

Learn how to add a Jekyll theme to your site.

Your site was last deployed to the github-pages environment by the pages build and deployment workflow.
Learn more about deploying to GitHub Pages using custom workflows

Custom domain

Custom domain
Custom domains allow you to serve your site from a domain other than `dhotre12.github.io`. Learn more.



Conclusion:

Application (write two Application)

Questionnaire:

1. What type of site or app is best suited for a PWA?

2. What are some common misconceptions about PWAs?

3. Can I use web components when building PWAs?

4. Do I need to build a single page app (SPA) before I build a PWA?

5. What frameworks work best for building a PWA?