The Destini Challenge

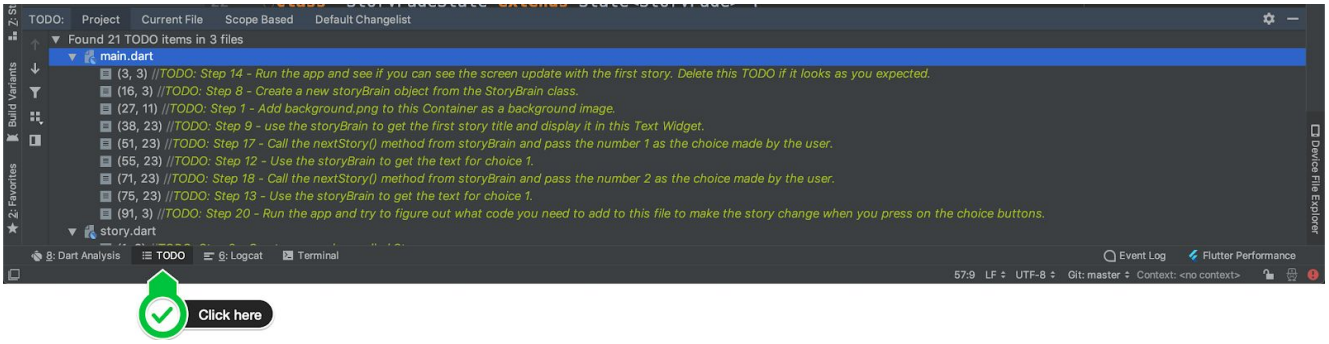
Set Up the Project

We'll be working from a starting project that has all the required assets and starting code. Head over to the GitHub page to clone it to your local system and click on "Get Dependencies" to get rid of the errors.

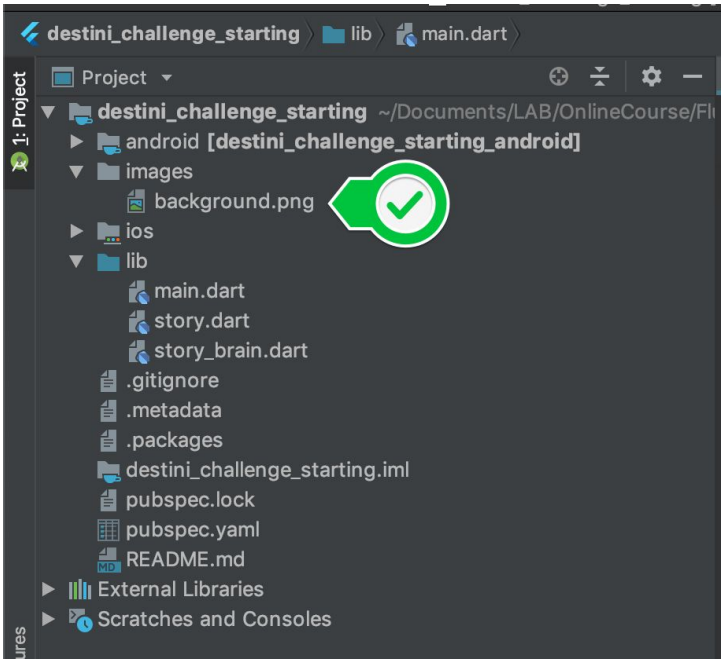
https://github.com/londonappbrewery/destini-challenge-starting

Familiarise Yourself with the Starting Project

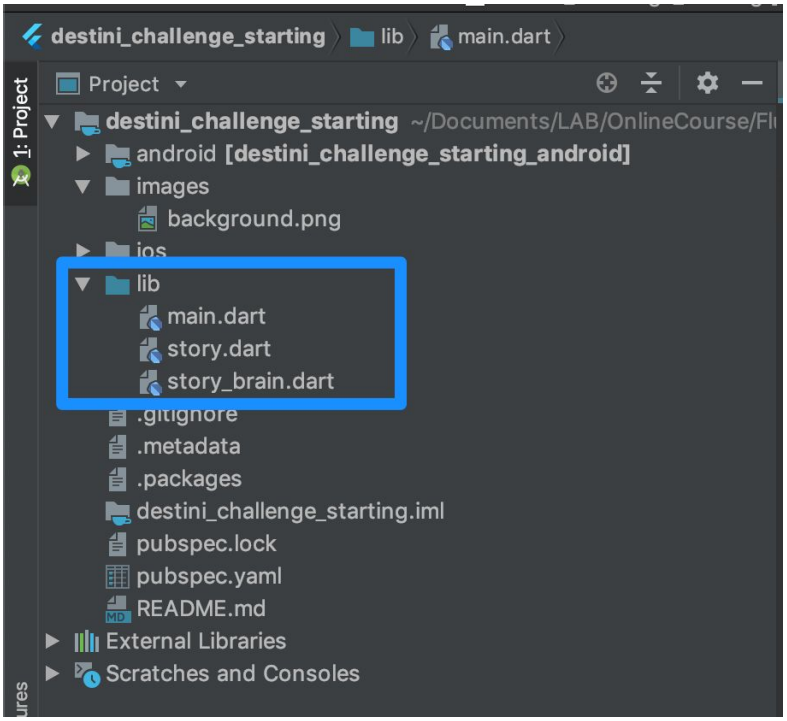
Take a look around the project. These are the most important points to notice:
1. A list of TODO items have been added to the project. You can view all of them by going to the TODO tab and complete them in order.



2. There is a folder called **images** and inside there is an image called **background.png**

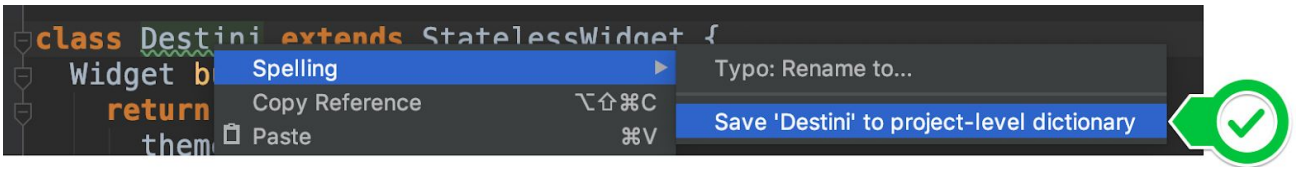


3. Inside the lib folder there are 3 files, **story.dart**, **story_brain.dart** and **main.dart**



4. Inside **main.dart** there is a Column Widget that contains a Text Widget and two FlatButton Widgets. **Run** the app to see how this looks in the app.

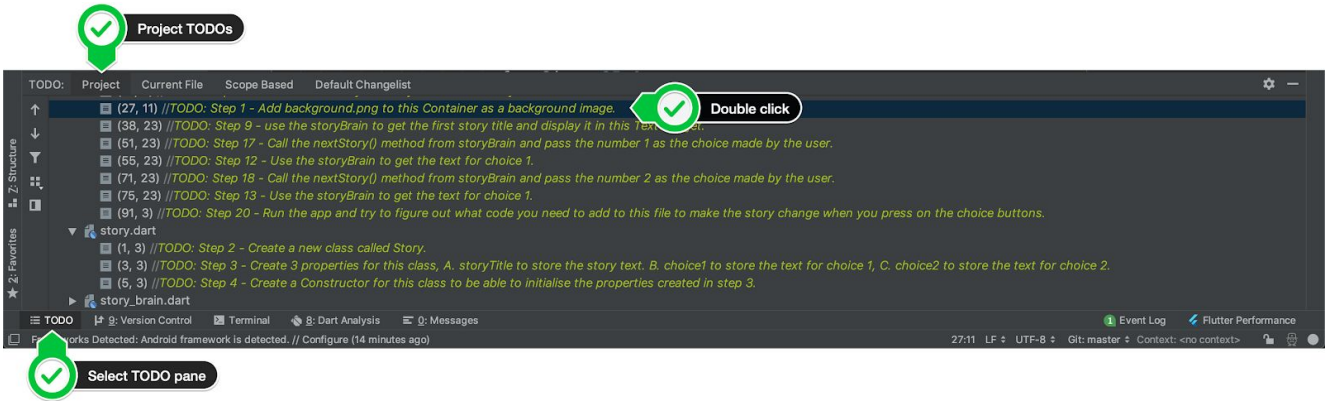
5. On line 7 of **main.dart** there's some squiggly lines under the word **Destini**, this is because it's not a dictionary word. If it bothers you, right click on it and go to Spelling --> Save 'Destini' to project-level dictionary.



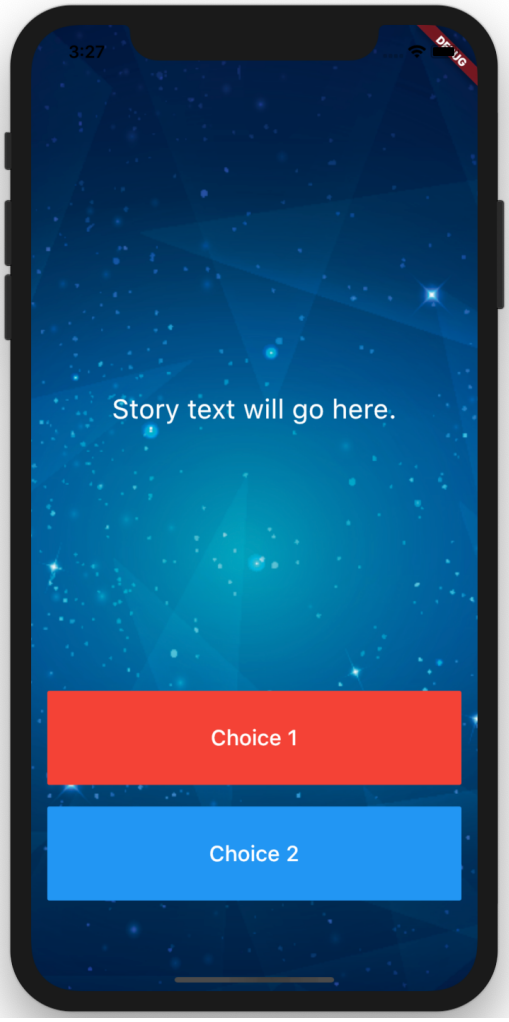
Code Challenges

Go into the TODOs pane, making sure that Project TODOs are showing and **double click** on Step 1. You should be taken to the first TODO in main.dart.

//TODO: Step 1 - Add background.png to this Container as a background image.



Using the Flutter docs and this StackOverflow post, figure out how to make the background filled by the background.png image.
NOTE: The pubspec.yaml has already been edited to add the images folder to the project as an asset source.
When you run the app, the end result should look like this:



If you really can't figure out how to complete the challenge, in every step, there is a link to the solution on GitHub. When you click on the link, you can see the solution on the right hand side. The + in front of the lines of code means that a new line of code was added compared to the last step.
Step 1 - SOLUTION

Solutions on GitHub

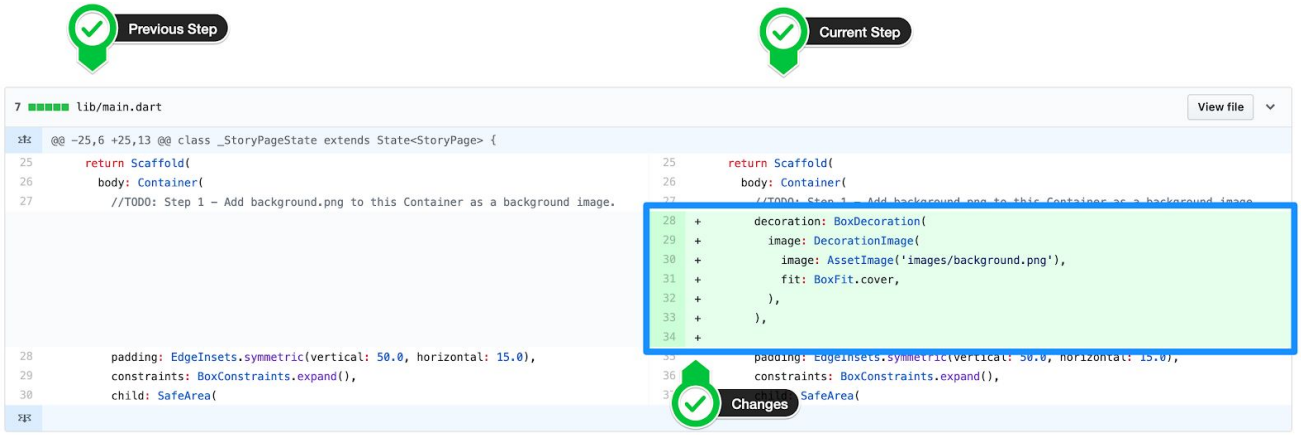
In this challenge, we'll be providing solutions for every step. This is so that if you really get stuck on one step you can check the solution and still continue with the rest of the challenges.
It's important that you try your hardest to complete the challenges, use the Flutter/Dart docs, Google and StackOverflow to really try and complete the steps by yourself. The tutorials are helpful for students to learn a topic, but it's completing the challenges that takes you closer to **mastery**!
The solutions are provided via Github commits.
A new line added looks like this:

```
3  +  //TODO: Step 3 - Create 3 properties for this class, A. storyTitle to store the story text. B. choice1 to store the text for choice 1, C. choice2 to store the text for choice
```

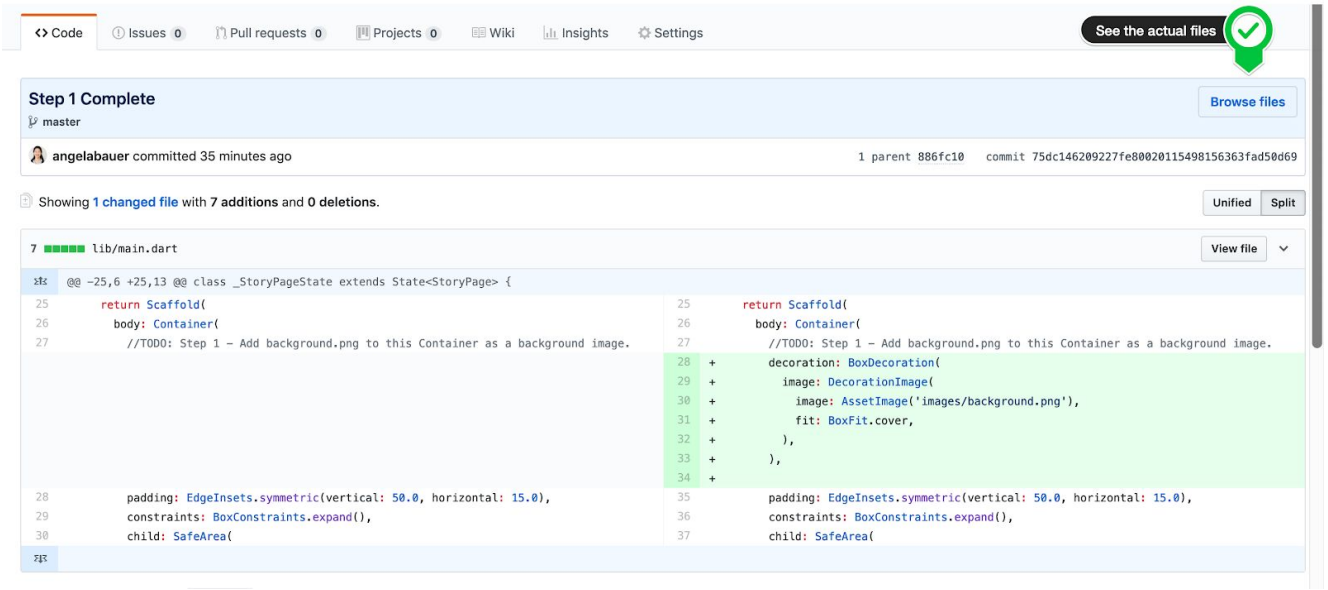
A line removed looks like this:

```
4  -//TODO: Step 3 - Create 3 properties for this class, A. storyTitle to store the story text. B. choice1 to store the text for choice 1, C. choice2 to store the text for choice 2.
```

REMEMBER, the solution is always on the **right hand pane**, the left hand pane is what the code looked like on the last step, in the right hand pane are the changes that completed the current step (the solution).



If you're confused how GitHub comments work, you can also just look at the files by clicking on browse files when you navigate to the solution.



//TODO: Step 2 - Create a new class called Story.

Using what you've learnt about classes and objects in the Quizzler module, create a **new** class called Story in story.dart.

If you need to review the docs on Dart Classes, you can find it here:

https://www.dartlang.org/guides/language/language-tour#classes

If you still can't figure out the solution, be sure to review the lessons on Dart Classes in the Quizzler module.

Step 2 - SOLUTION

//TODO: Step 3 - Create 3 properties for this class, A. storyTitle to store the story text. B. choice1 to store the text for choice 1, C. choice2 to store the text for choice 2.
HINT: These properties need to be created **inside** the Story class.
Properties are variables associated with a class, they are also often referred to as a **field** or an **instance variable.**
https://www.dartlang.org/guides/language/language-tour#instance-variables
Step 3 - SOLUTION

//TODO: Step 4 - Create a Constructor for this class to be able to initialise the properties created in step 3.
If you need a reminder about Constructors, have a look at the Dart docs here:
https://www.dartlang.org/guides/language/language-tour#constructors
Step 4 - SOLUTION

//TODO: Step 5 - Create a new class called StoryBrain.
This should be easy, you've already done it in step 2.
Step 5 - SOLUTION

//TODO: Step 6 - import the story.dart file into this file.
Step 6 - SOLUTION

//TODO: Step 7 - Uncomment the lines below to include storyData as a private property in StoryBrain.
Hint: You might need to import something to make this work.
Remember, the shortcut for uncommenting multiple lines in Android Studio is command + / for mac, and control + / for windows.
Step 7 - SOLUTION

//TODO: Step 8 - Create a method called getStory() that returns the first storyTitle from _storyData.
If you need to refresh yourself on how methods are declared head over here:
https://www.dartlang.org/guides/language/language-tour#methods
Step 8 SOLUTION

//TODO: Step 9 - Create a new storyBrain object from the StoryBrain class.
Heading back into the main.dart file, we'll need to use the StoryBrain class to create a new storyBrain object to work with.
Step 9 SOLUTION

//TODO: Step 10 - use the storyBrain to get the first story title and display it in this Text Widget.
Instead of having the hard coded 'Story text will go here.', use the storyBrain to provide the story title to display here.
Step 10 SOLUTION

//TODO: Step 11 - Create a method called getChoice1() that returns the text for the first choice1 from _storyData.
Step 11 SOLUTION

//TODO: Step 12 - Create a method called getChoice2() that returns the text for the first choice2 from _storyData.
Step 12 SOLUTION

//TODO: Step 13 - Use the storyBrain to get the text for choice 1.
Step 13 SOLUTION

//TODO: Step 14 - Use the storyBrain to get the text for choice 1.
Step 14 SOLUTION

//TODO: Step 15 - Run the app and see if you can see the screen update with the first story.

//TODO: Step 16 - Create a property called storyNumber which starts with a value of 0. This will be used to track which story the user is currently viewing.
Step 16 SOLUTION

//TODO: Step 17 - Create a method called nextStory(), it should not have any outputs but it should have 1 input which will be the choice number (int) made by the user.
Leave the body of the method empty, we'll get to it later.
Step 17 SOLUTION

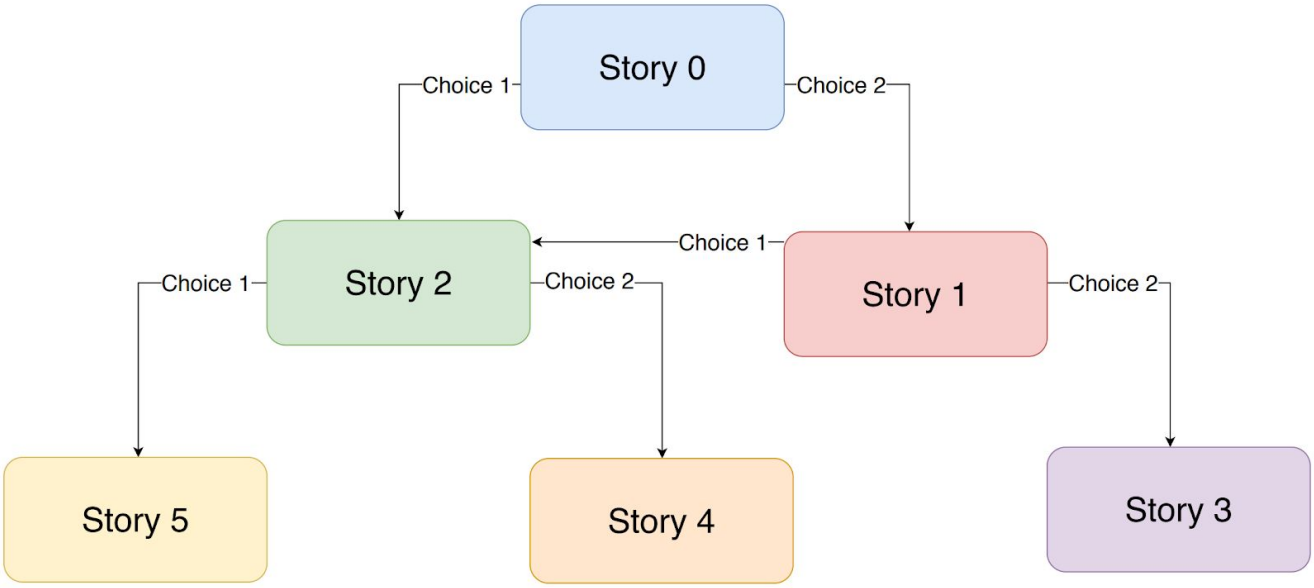
//TODO: Step 18 - Call the nextStory() method from storyBrain and pass the number 1 as the choice made by the user.
Step 18 SOLUTION

//TODO: Step 19 - Call the nextStory() method from storyBrain and pass the number 2 as the choice made by the user.
Step 19 SOLUTION

NOTE: At this stage our app won't do anything yet. We'll test it in the next challenge lesson.

//TODO: Step 20 - Download the story plan
Head over here to download the story plan:
https://drive.google.com/uc?export=download&id=1KU6EghkO9Hf2hRM0756xFHgNaZyGCou3

//TODO: Step 21 - Using the story plan, update nextStory to change the storyNumber depending on the choice made by the user.



This plan will determine how the user will progress through the story. If you're not familiar with how to choose your own adventure games work, be sure to read it up here:
https://en.wikipedia.org/wiki/Choose_Your_Own_Adventure

The user will start on story 0, the first story in the _storyData List.
If they click on the top button (red), they made Choice1 and the screen should refresh to show Story 2 and 2 new choices. But if they picked Choice 2, it should take them to story 1 instead.

This is how the whole story will play out at a glance:
If the user chooses choice 1 when they see Story0, they will be taken to Story2, from there, if they choose choice1 again, they get taken to story5 and the game ends there.

Take a look at the Story Outline on the next page.

Story Outline

Story0: Your car has blown a tire on a winding road in the middle of nowhere with no cell phone reception. You decide to hitchhike. A rusty pickup truck rumbles to a stop next to you. A man with a wide brimmed hat and soulless eyes opens the passenger door for you and says: "Need a ride, boy?".

Choice1: I'll hop in. Thanks for the help!

Story2: As you begin to drive, the stranger starts talking about his relationship with his mother. He gets angrier and angrier by the minute. He asks you to open the glove box. Inside you find a bloody knife, two severed fingers, and a cassette tape of Elton John. He reaches for the glove box.

Choice1: I love Elton John! Hand him the cassette tape.

Story5: You bond with the murderer while crooning verses of "Can you feel the love tonight". He drops you off at the next town. Before you go he asks you if you know any good places to dump bodies. You reply: "Try the pier."

Choice2: It's him or me. Take the knife and stab him.

Story6: As you smash through the guardrail and careen towards the jagged rocks below you reflect on the dubious wisdom of stabbing someone while they are driving a car you are in.

Choice2: Well, I don't have many options. Better ask him if he's a murderer.

Story1: He nods slowly, unphased by the question.

Choice1: At least he's honest. I'll climb in.

Story2: As you begin to drive, the stranger starts talking about his relationship with his mother. He gets angrier and angrier by the minute. He asks you to open the glove box. Inside you find a bloody knife, two severed fingers, and a cassette tape of Elton John. He reaches for the glove box.

Choice1: I love Elton John! Hand him the cassette tape.

Story5: You bond with the murderer while crooning verses of "Can you feel the love tonight". He drops you off at the next town. Before you go he asks you if you know any good places to dump bodies. You reply: "Try the pier."

Choice2: It's him or me. Take the knife and stab him.

Story4: As you smash through the guardrail and careen towards the jagged rocks below you reflect on the dubious wisdom of stabbing someone while they are driving a car you are in.

Choice2: Wait, I know how to change a tire.

Story4: What? Such a cop out! Did you know accidental traffic accidents are the second leading cause of accidental death for most adult age groups?

You can download this story outline as a PDF here: https://drive.google.com/uc?export=download&id=1XgnMqvQ7AldYUM-wS0wX5rDNT_GwxSjU

HINT: You'll need to use what you've learn about Dart conditionals (IF/ELSE IF/ELSE) to map out different pathways for the user.

Step 21 SOLUTION

So far, the nextStory method covers what should happen if the user needs to progress through the story. But what should happen when they reach one of the end points of the story, namely story 3, 4 or 5?

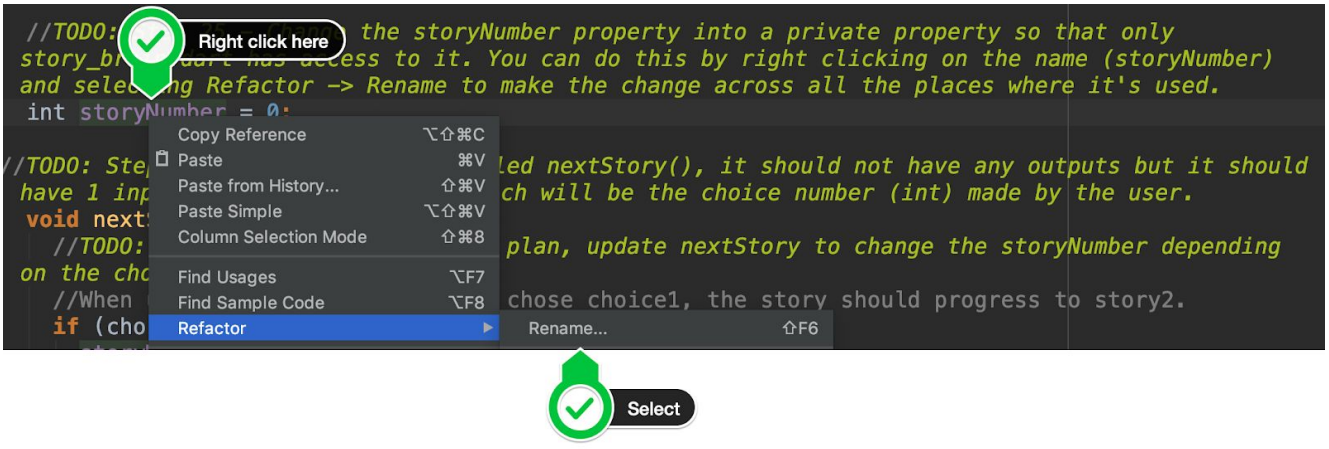
//TODO: Step 22 - In nextStory() if the storyNumber is equal to 3 or 4 or 5, that means it's the end of the game and it should call a method called restart() that resets the storyNumber to 0.
HINT: You'll need to add some conditions to the nextStory() method to achieve this.
Step 22 SOLUTION

//TODO: Step 23 - Use the storyNumber property inside getStory(), getChoice1() and getChoice2() so that it gets the updated story and choices rather than always just the first (0th) one.
Now that we have a storyNumber that tracks where the user is at in the storyline, it's time to put it to use in the methods that get the information for our screen. So instead of always just showing the 0th story, first choice and second choice, we need to update it using the storyNumber.
Step 23 SOLUTION

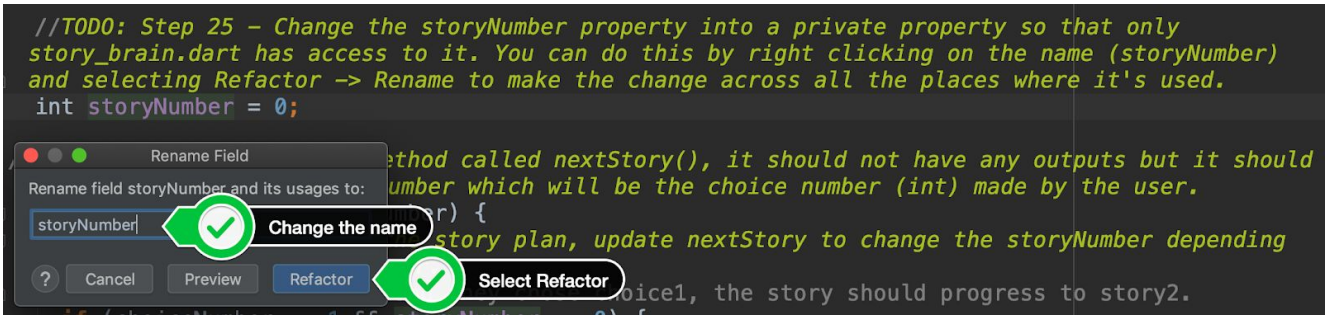
//TODO: Step 24 - Run the app and try to figure out what code you need to add to this file to make the story change when you press on the choice buttons.
If you run your app at this stage, you'll see that nothing happens when you press on the choice buttons. Why is this?
HINT
Step 24 SOLUTION

Android Studio makes it really easy for you to rename things that are scattered all over your project. In our story_brain.dart we want to make the storyNumber into a private property. But storyNumber is used in a lot of places. Without Android Studio, we would have to change it manually in over 10 places. But let's see what we can do using Android Studio's refactoring capabilities.

//TODO: Step 25 - Change the storyNumber property into a private property so that only story_brain.dart has access to it. You can do this by right clicking on the name (storyNumber) and selecting Refactor -> Rename to make the change across all the places where it's used.



Now when you rename the property, the changes will be made to every storyNumber that is used in the project.

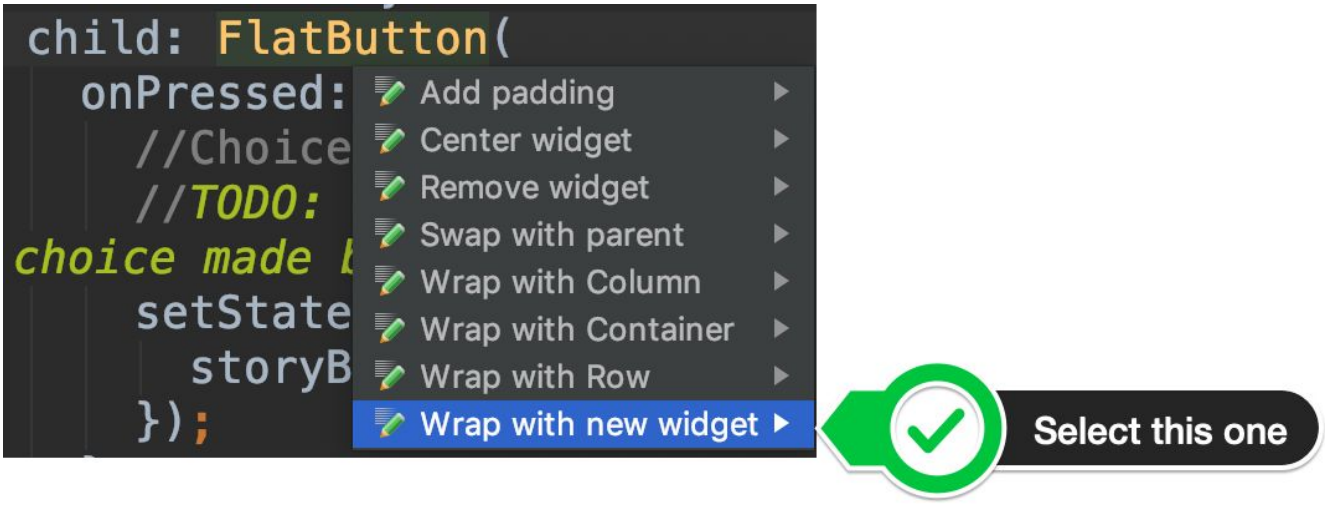


HINT

Step 25 SOLUTION

If you run the app now, everything should work as expected and the app has all the functionality it needs. But when the user reaches the end of the storyline, the top button reads Restart, but the bottom button doesn't say anything. There is no second choice for the user to make. So it makes more sense to hide the second button. But how do we do this using Flutter?
Let's google this: https://www.google.com/search?q=flutter+hide+widget

//TODO: Step 26 - Use a Flutter Visibility Widget to wrap this FlatButton.
Remember, in order to wrap a Widget in another Widget, you can use the light bulb or the **Show Intention Actions** shortcut:
mac: ⌥ + ↩
windows: Alt + Enter



HINT
Step 26 SOLUTION

//TODO: Step 27 - Create a method called buttonShouldBeVisible() which checks to see if storyNumber is 0 or 1 or 2 (when both buttons should show choices) and return true if that is the case, else it should return false.
We need a way of checking if at this current point in the storyline if the bottom blue button should be visible or should be hidden. We'll create a method in the story_brain.dart to do this.
Step 27 SOLUTION

//TODO: Step 28 - Set the "visible" property of the Visibility Widget to equal the output from the buttonShouldBeVisible() method in the storyBrain.
The Visibility Widget accepts a property called visible. When this is set to true, the child of the Visibility Widget will be visible, if it's false, it will remove the child Widget from the screen.
Step 28 SOLUTION

//TODO: Step 29 - Run the app and test it against the Story Outline to make sure you've completed all the steps. The code for the completed app can be found here: https://github.com/londonappbrewery/destini-challenge-completed/
If you need to check your code against the completed app, you can find it here:
https://github.com/londonappbrewery/destini-challenge-completed/
Feel free to git clone it to your local system and compare it against your code if you have any niggling bugs you haven't managed to solve.

More Dart Programming Challenges

If you are already familiar with OOP and want to quickly get up to speed with the Dart programming language. I recommend heading over to the Dart Track on Exercism and completing the track as well as extra exercises.

Or

If you are completely new to programming and want to test yourself as you progress through the course, I recommend starting with the first 3 exercises in the Dart Track on Exercism and progressing as your capability grows. This might mean coming back to Exercism when you learn more Dart concepts through the course.

Dart Track

Joined on 2nd February 2019

Hello World

control flow conditionals · optional values

IN PROGRESS

The classical introductory exercise. Just say "Hello, World!"

Two Fer

conditionals · optional values

LOCKED

Create a sentence of the form "One for X, one for me."

Leap

booleans · integers · logic

LOCKED

Given a year, report if it is a leap year.

Scrabble Score

control flow conditionals · control flow loops

LOCKED

Given a word, compute the scrabble score for that word.