

Assignment
Mobile Application Development
SAYYAD AMIN
SP21-BCS-015

Tutorial for Dart Language:

Table of Contents:

1. Introduction to Dart

- Overview of Dart
- History of Dart
- Features of Dart
- Advantages of using Dart

2. Getting Started with Dart

- Installing Dart
- Hello World in Dart
- Dart development tools
- Dart syntax

3. Variables and Data Types

- Variables in Dart
- Primitive Data Types in Dart
- Strings in Dart
- Numbers in Dart

4. Control Flow and Loops

- If-else statements
- Switch statements
- For loops
- While loops

5. Functions and Methods

- Defining functions in Dart
- Parameters and arguments

- Return values
- Anonymous functions

6. Classes and Objects

- Object-oriented programming in Dart
- Classes and objects
- Constructors
- Inheritance and polymorphism

7. Collections

- Lists in Dart
- Maps in Dart
- Sets in Dart

8. Exception Handling

- Handling errors in Dart
- Try-catch blocks
- Throwing exceptions

9. Asynchronous Programming

- Asynchronous programming in Dart
- Futures and async/await
- Streams and asynchronous data processing

10. Libraries and Packages

- Creating libraries
- Importing and using libraries
- Creating and publishing packages

11. Web Programming with Dart

- Introduction to web programming with Dart
- Creating web applications with Dart
- Using Dart with web frameworks like Flutter and AngularDart

12. Conclusion

- Summary of Dart features and capabilities
- Future of Dart programming language.

Introduction to Dart:

Google created Dart, a contemporary object-oriented programming language. It is designed to be fast, flexible, and easy to learn. Dart can be used to create web, mobile, and desktop applications. In this tutorial, we will cover the basics of the Dart programming language.

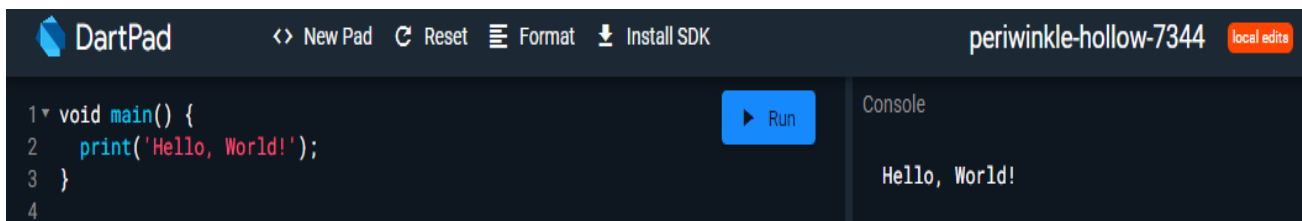
Getting Started with Dart

Before we can start writing Dart code, we need to install the Dart SDK. The SDK includes the Dart programming language, libraries, and tools. You can download the SDK from the Dart website. Once you have installed the SDK, you can start writing Dart code using a text editor or an Integrated Development Environment (IDE) such as Visual Studio Code or IntelliJ IDEA.

Hello World in Dart:

Let's start by writing a simple "Hello, World!" program in Dart. Open your text editor or IDE and create a new file called "hello.dart." Then, add the following code:

Hello world:

A screenshot of the DartPad web IDE interface. The top bar shows the DartPad logo, navigation links like 'New Pad', 'Reset', 'Format', and 'Install SDK', a user identifier 'periwinkle-hollow-7344', and a 'local edit' button. The main editor area contains Dart code:

```
1 void main() {  
2   print('Hello, World!');  
3 }  
4
```

 A blue 'Run' button is positioned to the right of the code. On the right side, there is a 'Console' panel displaying the output 'Hello, World!'.

Variables and data types are fundamental concepts in programming, and Dart is no exception. In Dart, variables are used to store data, and data types define the kind of data that can be stored in a variable.

Variables:

In Dart, we use the "var" keyword before the variable name and, if we want, an initial value. The syntax for declaring a variable in Dart is as follows:

Here, "variableName" is the name of the variable, and "initialValue" is an optional initial value that can be assigned to the variable.

Data Types:

Dart has several built-in data types for storing different kinds of data:

- **numbers: int and double**
- **strings: String**
- **booleans: bool**
- **lists: List**
- **maps: Map**
- **sets: Set**

Numbers:

Dart has two built-in numeric data types: int and double. Integers are whole numbers with no fractional part, while doubles are floating-point numbers with a fractional part. In Dart, both int and double are subtypes of num, which is the base type for numeric values.

Strings:

Strings are sequences of characters enclosed in quotes. Dart supports both single and double quotes for defining strings. In Dart, strings are immutable, which means that once a string is created, it cannot be modified.

Booleans:

Booleans represent logical values that can be either true or false. In Dart, the bool data type is used to represent boolean values.

Lists:

Lists are ordered collections of objects. In Dart, lists can contain objects of any type, including other lists. Lists in Dart are mutable, which means that their contents can be modified after they are created.

Maps:

Maps are unordered collections of key-value pairs. In Dart, keys and values can be of any type, and maps can contain other maps as values. Maps in Dart are mutable, which means that their contents can be modified after they are created.

Sets:

Sets are unordered collections of unique objects. In Dart, sets can contain objects of any type, but each object can appear in the set only once. Sets in Dart are mutable, which means that their contents can be modified after they are created.


Understanding variables and data types is essential for writing effective and efficient Dart code. By giving each variable the right data type, you can make sure that your code is correct and works well.

Control Flow:

Control flow statements in Dart let us decide how our program should run based on certain conditions. The most common control flow statement is the "if-else" statement. It allows us to execute a certain block of code if a condition is true and a different block of code if the condition is false.


Another control flow statement is the "switch" statement. It allows us to execute different blocks of code based on the value of a variable.

If-else statement:

 DartPad <> New Pad ↺ Reset ≡

```
1 ▾ void main() {  
2   int x = 10;  
3  
4   // If statement  
5 ▾ if (x > 0) {  
6     print('x is positive');  
7   }  
8  
9   // If-else statement  
10 ▾ if (x % 2 == 0) {  
11     print('x is even');  
12 ▾ } else {  
13     print('x is odd');  
14   }  
15  
16   // Switch statement  
17 ▾ switch (x) {  
18     case 0:  
19       print('x is zero');  
20       break;  
21     case 10:  
22       print('x is ten');  
23       break;  
24     default:  
25       print('x is not zero or ten');  
26   }  
27 }  
28
```

Result:


Console

```
x is positive  
x is even  
x is ten
```

Loops:

Dart's loops let us iterate over groups of data or run a block of code more than once. The most common loop is the "for" loop. It lets us run a block of code a certain number of times, based on how long a collection is or how many times we tell it to run.

Another loop is the "while" loop. It allows us to execute a block of code while a certain condition is true. The "do-while" loop is similar to the "while" loop, but it always executes the block of code at least once, regardless of whether the condition is true.

Loops:



The screenshot shows the DartPad web interface. The code editor contains the following Dart code:

```
1 void main() {  
2   // For loop  
3   for (int i = 1; i <= 10; i++) {  
4     print(i);  
5   }  
6  
7   // While loop  
8   int j = 1;  
9   while (j <= 10) {  
10    print(j);  
11    j++;  
12  }  
13  
14  // Do-while loop  
15  int k = 1;  
16  do {  
17    print(k);  
18    k++;  
19  } while (k <= 10);  
20 }
```

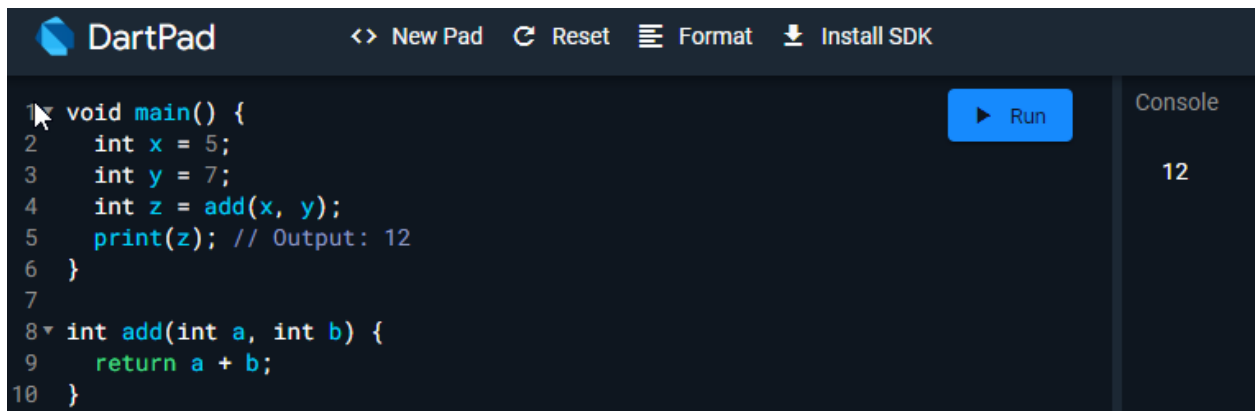
The console on the right shows the output of the code, which is the numbers 1 through 10, each on a new line. The interface also includes a 'Run' button and a 'Documentation' tab.

Functions:

Functions wrap up code that can be used more than once and make it easier to read and keep up-to-date. Functions wrap up code that can be used more than once and make it easier to read and keep up-to-date. "returnType" is the type of value that the function returns, or "void" if it does not return a value.

- "functionName" is the name of the function.
- "parameters" are the input values that the function expects, enclosed in parentheses.

function:



```
DartPad <> New Pad ↺ Reset ≡ Format ⬇ Install SDK

1 void main() {
2   int x = 5;
3   int y = 7;
4   int z = add(x, y);
5   print(z); // Output: 12
6 }
7
8 int add(int a, int b) {
9   return a + b;
10 }
```

Run

Console

12

Methods:

In object-oriented programming, a "method" is a function that is associated with an object or a class. Methods are used to wrap up behavior that is unique to a class or an object. "returnType" is the type of value that the method returns, or "void" if it does not return a value.

- "methodName" is the name of the method.
- "parameters" are the input values that the method expects, enclosed in parentheses.

In this example, the "Rectangle" class has two properties, "width" and "height," and a method called "area" that calculates the area of the rectangle by multiplying its width and height.

When you create an instance of a class, you can call its methods using the dot notation. In this example, we make an instance of the "Rectangle" class and set its width and

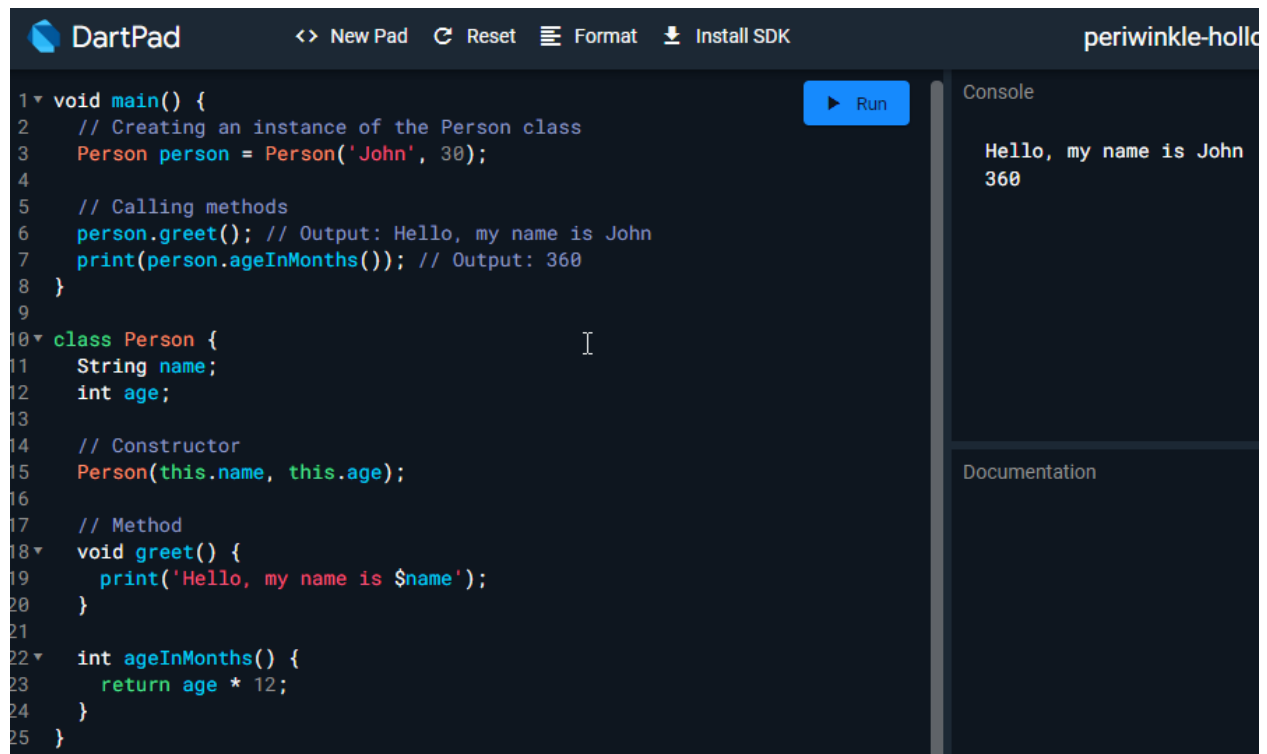
height properties. We then call the "area" method using the dot notation and print the result to the console.

Classes and Objects:

Dart classes are the blueprint of the object, or they can be called object constructors. A class can contain fields, functions, constructors, etc. It is a container that holds the data and functions together. You can get to it by making an object. A class can be used to talk about a user-defined data type that gives all of its objects the same traits.

We can assume a class as a sketch (prototype) or a car. It contains all the details about the model name, year, features, price, etc. Based on these properties, we can build the car. Here, the car is an object. There can be many cars, so we can create many objects of cars to access all the properties.

Classes:



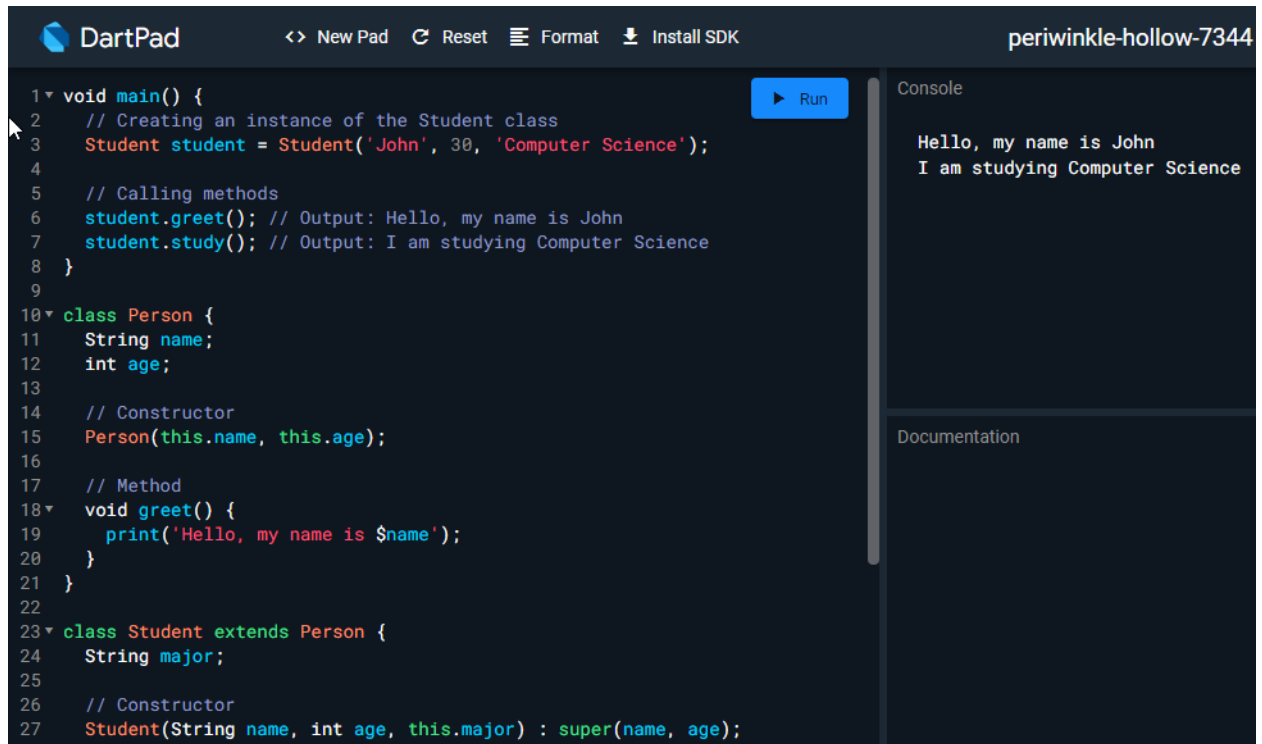
```
1 void main() {
2   // Creating an instance of the Person class
3   Person person = Person('John', 30);
4
5   // Calling methods
6   person.greet(); // Output: Hello, my name is John
7   print(person.ageInMonths()); // Output: 360
8 }
9
10 class Person {
11   String name;
12   int age;
13
14   // Constructor
15   Person(this.name, this.age);
16
17   // Method
18   void greet() {
19     print('Hello, my name is $name');
20   }
21
22   int ageInMonths() {
23     return age * 12;
24   }
25 }
```

Console

```
Hello, my name is John
360
```

Documentation

Inheritance:



DartPad

<> New Pad ↺ Reset ≡ Format ⬇ Install SDK

periwinkle-hollow-7344

```
1 void main() {
2   // Creating an instance of the Student class
3   Student student = Student('John', 30, 'Computer Science');
4
5   // Calling methods
6   student.greet(); // Output: Hello, my name is John
7   student.study(); // Output: I am studying Computer Science
8 }
9
10 class Person {
11   String name;
12   int age;
13
14   // Constructor
15   Person(this.name, this.age);
16
17   // Method
18   void greet() {
19     print('Hello, my name is $name');
20   }
21 }
22
23 class Student extends Person {
24   String major;
25
26   // Constructor
27   Student(String name, int age, this.major) : super(name, age);
28 }
```

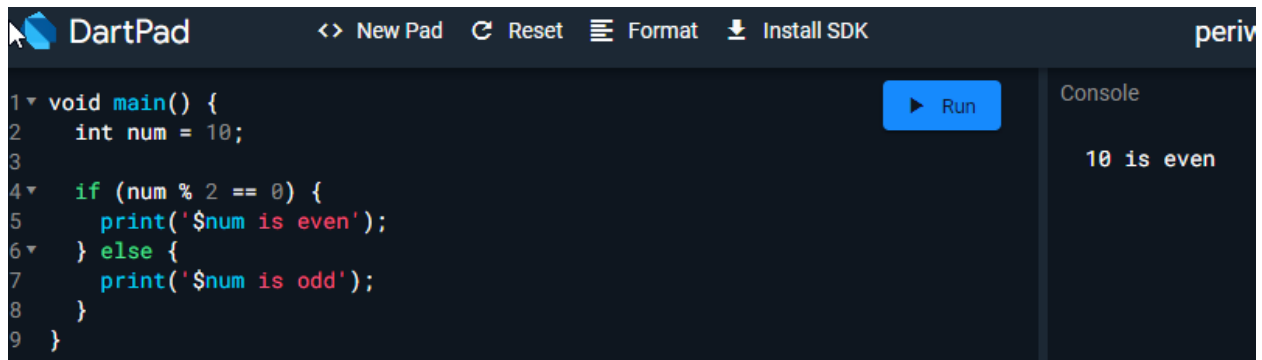
Run

Console

Hello, my name is John
I am studying Computer Science

Documentation

Even Odd:



DartPad

<> New Pad ↺ Reset ≡ Format ⬇ Install SDK

periv

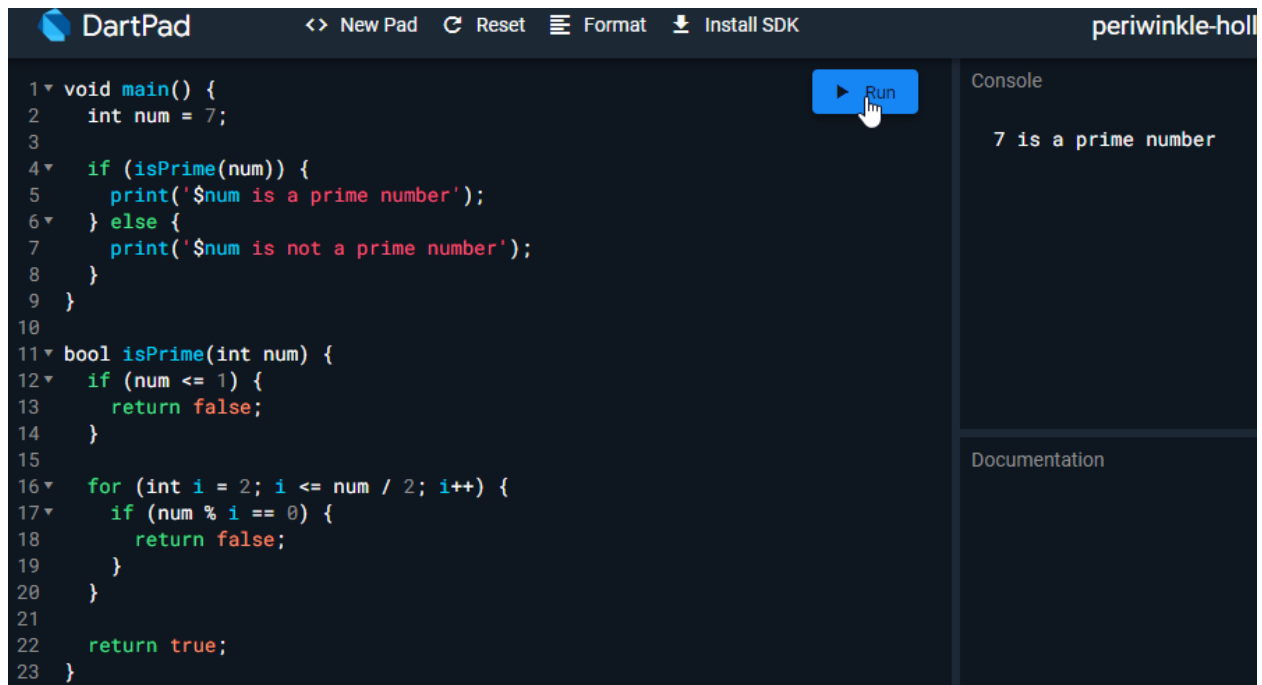
```
1 void main() {
2   int num = 10;
3
4   if (num % 2 == 0) {
5     print('$num is even');
6   } else {
7     print('$num is odd');
8   }
9 }
```

Run

Console

10 is even

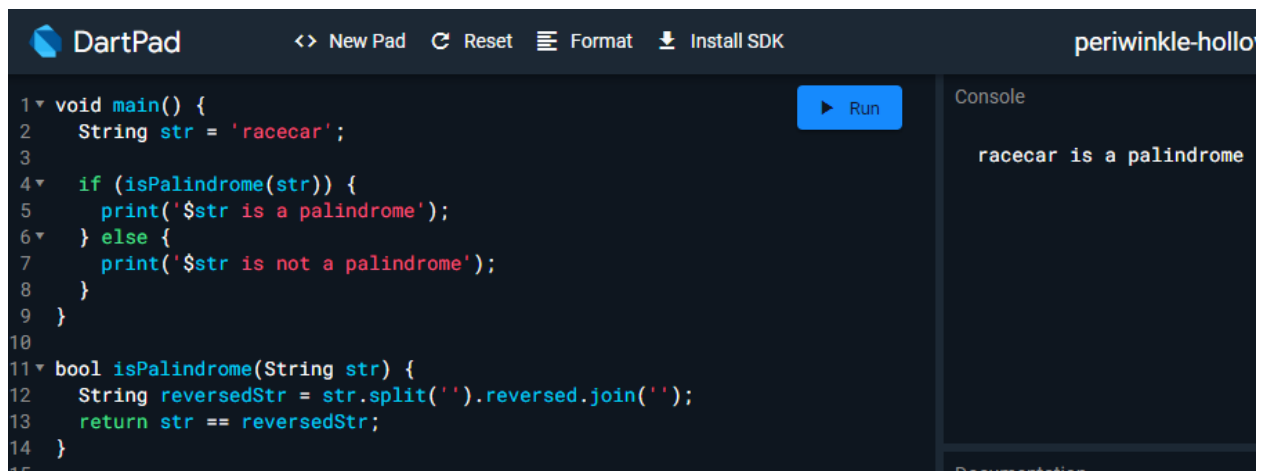
Prime Number:



DartPad interface showing Dart code for a prime number checker. The code defines a `main` function and an `isPrime` function. The console output shows "7 is a prime number".

```
1 void main() {  
2   int num = 7;  
3  
4   if (isPrime(num)) {  
5     print('$num is a prime number');  
6   } else {  
7     print('$num is not a prime number');  
8   }  
9 }  
10  
11 bool isPrime(int num) {  
12   if (num <= 1) {  
13     return false;  
14   }  
15  
16   for (int i = 2; i <= num / 2; i++) {  
17     if (num % i == 0) {  
18       return false;  
19     }  
20   }  
21  
22   return true;  
23 }
```

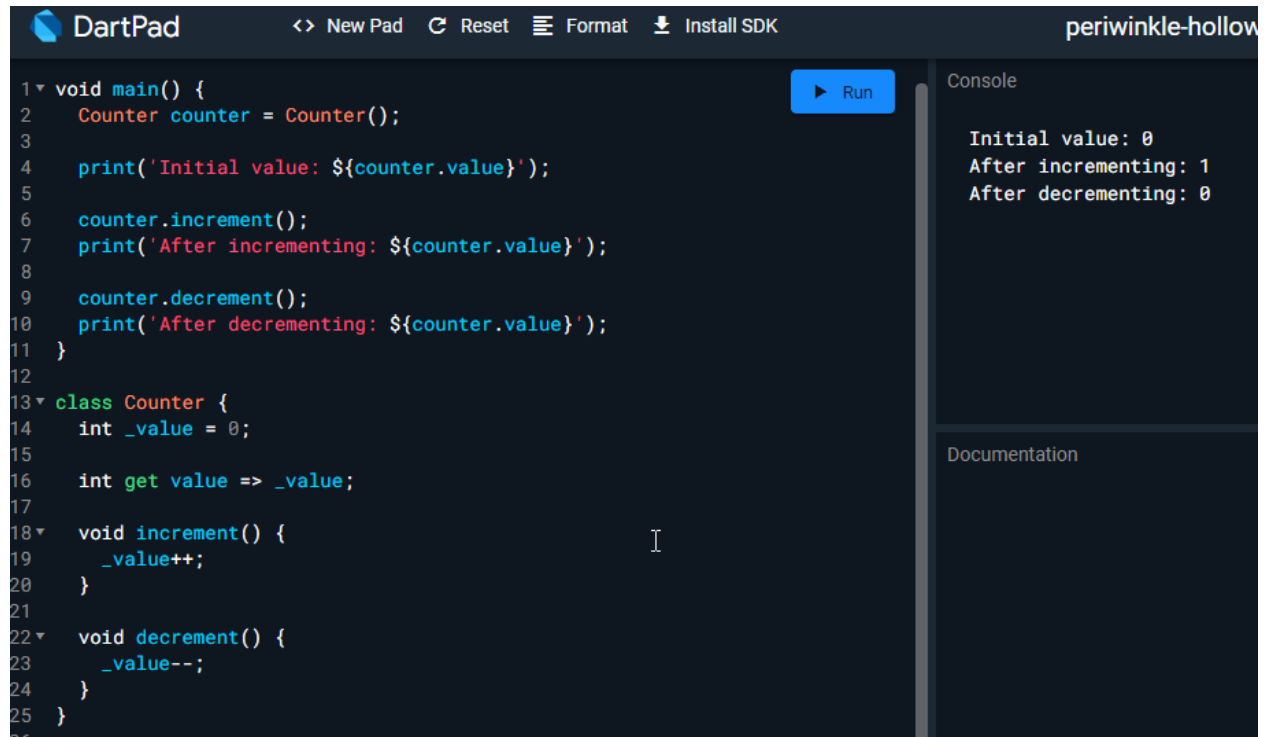
Palindrome Number:



DartPad interface showing Dart code for a palindrome checker. The code defines a `main` function and an `isPalindrome` function. The console output shows "racecar is a palindrome".

```
1 void main() {  
2   String str = 'racecar';  
3  
4   if (isPalindrome(str)) {  
5     print('$str is a palindrome');  
6   } else {  
7     print('$str is not a palindrome');  
8   }  
9 }  
10  
11 bool isPalindrome(String str) {  
12   String reversedStr = str.split('').reversed.join('');  
13   return str == reversedStr;  
14 }  
15
```

Counter Example:



DartPad

<> New Pad ↺ Reset ≡ Format ⬇ Install SDK

periwinkle-hollow

```
1 void main() {
2   Counter counter = Counter();
3
4   print('Initial value: ${counter.value}');
5
6   counter.increment();
7   print('After incrementing: ${counter.value}');
8
9   counter.decrement();
10  print('After decrementing: ${counter.value}');
11 }
12
13 class Counter {
14   int _value = 0;
15
16   int get value => _value;
17
18   void increment() {
19     _value++;
20   }
21
22   void decrement() {
23     _value--;
24   }
25 }
```

Run

Console

Initial value: 0
After incrementing: 1
After decrementing: 0

Documentation

Counter App After Some Modifications:

