socrative
A showbie PRODUCT

Name _____
Date _____

Flutter

Score _____

1. Which platforms does Flutter support for app deployment?

(A) Android
(B) iOS
(C) Web
(D) All of the above

2. What is the core building block of Flutter's architecture?

(A) Widgets
(B) Modules
(C) Classes
(D) Components

3. What is Dart primarily used for in Flutter?

(A) User Interface design
(B) Business Logic
(C) Networking
(D) Database operations

4. Which of the following are Dart data types?

(A) int
(B) double
(C) String
(D) bool
(E) List

5. Which IDE is commonly used for Flutter development?

(A) Visual Studio Code

(B) IntelliJ IDEA

(C) Android Studio

(D) All of the answers

(E) Eclipse

6. What is the purpose of async/await functions in Dart?

(A) To handle asynchronous operations

(B) To define classes

(C) To create widgets

(D) To manage databases

7. What is the primary purpose of the lib folder in a Flutter project?

(A) Storing images

(B) Managing dependencies

(C) Organizing source code files

(D) Defining routes

8. What is the purpose of the Widget Tree and Element Tree in Flutter?

(A) Managing app state

(B) Building the user interface

(C) Handling exceptions

(D) Defining routes

9. How do you handle asynchronous programming in Dart?

(A) Using Future and FutureBuilder

(B) Using Streams and StreamBuilder

(C) Using async/await functions

(D) All of the answers

(E) None of the answers

10. What is a route in Flutter?

(A) A widget

(B) A screen or page

(C) A function

(D) An exception

11. How do you pass data between screens in Flutter?

(A) Using global variables

(B) Using constructor parameters

(C) Using Navigator arguments

(D) No of those answers

12. What is Shared Preferences used for in Flutter?

(A) Managing app state

(B) Storing small pieces of data persistently

(C) Handling exceptions

(D) Making HTTP requests

13. Which of the following is a local database solution in Flutter?

(A) SQLite

(B) Firebase

(C) MongoDB

(D) Cassandra

14. How does Dart handle null safety?

(A) Using the ? operator

(B) Using the ! operator

(C) Using the ?? operator

15. What keyword is used to handle exceptions in Dart?

(A) try

(B) catch

(C) throw

(D) =>

16. How do you declare a single-line function in Dart?

(A) Using braces {} for the function body

(B) Using the => arrow

(C) Using the function keyword

(D) All of the above

17. What is the purpose of the async keyword in Dart functions?

(A) To declare a function as asynchronous

(B) To define a function as a stream

(C) To handle exceptions

(D) All of the above

18. How is a Future different from a regular value in Dart?

(A) A Future represents a value that may not be available yet

(B) A Future cannot be awaited

(C) A Future is a primitive data type

(D) A Future cannot be used in control flow

19. Choose the Dart code that declares a variable named `age` with the type integer.

(A) int age = 25;

(B) age := 25;

(C) var age = "25";

(D) age = 25;

20. Choose the Dart code that demonstrates proper null safety for a variable named `name`.

(A) String name = null;

(B) String? name = "John";

(C) String name = "John";

(D) String? name = null;

21. Identify the correct Dart code for handling an exception when dividing two numbers.

(A)
```
try {
  var result = num1 / num2;
} catch (e) {
  print("Error: $e");
}
```

(B)
```
var result = num1 / num2;
if (result.isNaN) {
  print("Error: Division by zero");
}
```

(C)
```
try {
  var result = num1 ~/ num2;
} on IntegerDivisionByZeroException {
  print("Error: Division by zero");
}
```

(D)
```
var result = num1 / num2;
throw Exception("Division by zero");
```

22. Choose the Dart code that defines a function named `sum` taking two parameters and returning their sum.

(A)
```
int sum(int a, int b) {
  return a * b;
}
```

(B)
```
int sum(int a, int b) => a + b;
```

(C)
```
int sum(a, b) {
  return a + b;
}
```

(D)
```
int sum(int a, int b) {
  return a - b;
}
```

23. Choose the Dart code that correctly defines an anonymous function assigned to the variable `square` that calculates the square of a given number.

(A)
```
var square = (int x) => x * x;
```

(B)
```
var square = (int x) {
  return x * x;
};
```

(C)
```
Function square = (int x) => x * x;
```

(D)
```
Function square = (int x) {
  return x * x;
};
```

24. Identify the Dart code that demonstrates a higher-order function.

(A)
```
int operate(int a, int b, Function operation) {
  return operation(a, b);
}
```

(B)
```
int operate(int a, int b, Function operation) {
  return a + b;
}
```

(C)
```
Function operate(int a, int b, Function operation) {
  return operation(a, b);
}
```

(D)
```
void operate(int a, int b, Function operation) {
  operation(a, b);
}
```

25. Choose the Dart code that correctly demonstrates the use of async/await for fetching data from an API.

(A)
```
Future<String> fetchData() {
  return http.get("api.example.com/data");
}
// Usage:
var data = await fetchData();
```

(B)
```
Future<String> fetchData() async {
  return http.get("api.example.com/data");
}

// Usage:
var data = fetchData();
```

(C)
```
Future<String> fetchData() {
  return await http.get("api.example.com/data");
}

// Usage:
var data = fetchData();
```

(D)
```
Future<String> fetchData() async {
  return await http.get("api.example.com/data");
}

// Usage:
var data = await fetchData();
```