**1. What Are Variables in Dart?**

In Dart, a variable is a named reference to an object stored in memory. Unlike some languages where variables might directly hold primitive values, Dart variables hold references. This means that even when a variable seems to store a simple value (like an integer or a string), it actually points to an object that holds that value.

**2. Type Inference vs. Explicit Typing**

Dart is statically typed but supports both explicit type annotations and type inference:

* **Type Inference with var:** When you declare a variable with var, the compiler infers the type from the assigned value at compile time:

var name = 'Bob'; // Dart infers type String.

var age = 25;     // Dart infers type int.

Although concise, once inferred, the type cannot change. This means:

var name = 'Alice';

// name = 42; // Error: Cannot assign an int to a String variable.

**Explicit Typing:**

You may declare the type explicitly, which enhances code clarity, especially for public APIs or class members:

String greeting = 'Hello';

int year = 2025;

Object flexible = 'Can hold any object type';

Explicit typing is also useful when the initial value might be ambiguous or when you plan to use a broader type (like Object or dynamic) later.

Dart provides a rich type system that includes:

* **Numbers:** int, double, and num for numeric operations.
* **Boolean:** bool for true/false values.
* **String:** For text data.
* **Collections:** List, Set, and Map for various data grouping needs.
* **Runes:** To work with Unicode code points.
* **Symbols:** For identifier names used in reflection.
* **Dynamic:** For flexible types with runtime type changes.
* **Object:** As the base type for all Dart types.

**1. Numbers**

Dart has three main numeric types: **int**, **double**, and **num**.

**int:** Represents whole numbers (both positive and negative).  
           int wholeNumber = 42;

print('Int: $wholeNumber'); // Output: Int: 42

**double:**           Represents numbers with a decimal point (floating-point numbers).  
           double pi = 3.14159;

     print('Double: $pi'); // Output: Double: 3.14159

**num:**  A common superclass for both int and double. It allows you to store either type.  
            num someNumber = 10;     // Could be an int

     print('Num as int: $someNumber'); // Output: Num as int: 10

      someNumber = 10.5;       // Now a double

print('Num as double: $someNumber'); // Output: Num as double: 10.5

**2. Boolean**

The **bool** type represents a true/false value.

bool isDartFun = true;

print('Boolean: $isDartFun'); // Output: Boolean: true

Booleans are often used in control flow statements (if, while, etc.).

**3. Strings**

A **String** is a sequence of characters used to represent text.

String greeting = 'Hello, Dart!';

print('String: $greeting'); // Output: String: Hello, Dart!

Strings can be manipulated using various built-in methods such as substring, split, and replace.

**4. Collections**

Dart provides several collection types:

**a. List**

A **List** is an ordered collection of items. You can specify the type of elements in the list.

Lists support indexing, iteration, and many useful methods (e.g., add, remove).

List<String> fruits = ['Apple', 'Banana', 'Cherry'];

print('List: $fruits'); // Output: List: [Apple, Banana, Cherry]

 print('List: ${fruits.join(', ')}'); // Output: List: Apple, Banana, Cherry

**b. Set**

A **Set** is an unordered collection of unique items. Duplicate entries are automatically removed.

Sets are useful when you need to ensure no duplicates are present.

Set<int> uniqueNumbers = {1, 2, 5, 3, 2}; //don't print repet ative values

  print('Set: $uniqueNumbers'); // Output: Set: {1, 2, 3}

  print('Set: ${uniqueNumbers.join(',')}');

**c. Map**

A **Map** is a collection of key-value pairs. The keys must be unique.

Maps are excellent for lookups and representing relationships between keys and values.

Map<String, int> ages = {

'Alice': 30,

'Bob': 25,

'Charlie': 35,

};

print('Map: $ages'); // Output: Map: {Alice: 30, Bob: 25, Charlie: 35}

**5. Runes**

**Runes** represent Unicode code points of a string. They are useful for handling characters outside the basic multilingual plane (e.g., emojis).

String heart = '❤️';//print decimal valure of characters

Runes runes = heart.runes;

print('Runes: ${runes.toList()}'); // Output: Runes: [10084, 65039]

Runes allow you to work with the numerical representation of Unicode characters.

**6. Symbols**

A **Symbol** represents an operator or identifier name. It is mainly used in APIs that rely on reflection.

Symbol mySymbol = #example;

print('Symbol: $mySymbol'); // Output: Symbol: Symbol("example")

Symbols are rarely used in everyday Dart programming but can be useful in advanced scenarios.

**7. Dynamic**

The **dynamic** type allows a variable to hold any type of value, and its type can change over time. However, using dynamic bypasses compile-time type checking.

dynamic variable = 'Hello, world!';

print('Dynamic (as String): $variable'); // Output: Dynamic (as String): Hello, world!

variable = 123;

print('Dynamic (as int): $variable'); // Output: Dynamic (as int): 123

Use **dynamic** sparingly as it can make code less predictable and harder to maintain.

**8. Object**

The **Object** type is the base type for all Dart objects. Every type in Dart is a subclass of Object.

Object obj = 'I am an object';

print('Object: $obj'); // Output: Object: I am an object

Even though **Object** is a supertype of all classes, explicit casting may be necessary when you need a more specific type.

**Full Example: Putting It All Together**

void main() {

// 1. Numbers

// Dart supports whole numbers (int), floating-point numbers (double), and a common type (num) that can hold either.

int wholeNumber = 42;

double pi = 3.14159;

num flexibleNumber = 10; // Initially an int

flexibleNumber = 10.5; // Now holds a double

print('Int: $wholeNumber'); // Output: Int: 42

print('Double: $pi'); // Output: Double: 3.14159

print('Num (int then double): $flexibleNumber'); // Output: Num (int then double): 10.5

// 2. Boolean

// The bool type represents true/false values and is commonly used in conditions.

bool isDartFun = true;

bool isLearning = false;

print('Boolean isDartFun: $isDartFun'); // Output: Boolean isDartFun: true

print('Boolean isLearning: $isLearning'); // Output: Boolean isLearning: false

// 3. Strings

// Strings are sequences of characters. You can use either single or double quotes.

String greeting = 'Hello, Dart!';

String message = "Dart is awesome.";

print('String greeting: $greeting'); // Output: String greeting: Hello, Dart!

print('String message: $message'); // Output: String message: Dart is awesome.

// 4. Collections

// Dart provides various collection types like List, Set, and Map.

// List: An ordered collection of items.

List<String> fruits = ['Apple', 'Banana', 'Cherry'];

print('List of fruits: $fruits'); // Output: List of fruits: [Apple, Banana, Cherry]

// Set: An unordered collection of unique items.

Set<int> uniqueNumbers = {1, 2, 3, 2};

print('Set of unique numbers: $uniqueNumbers'); // Output: Set of unique numbers: {1, 2, 3}

// Map: A collection of key-value pairs.

Map<String, int> ages = {

'Alice': 30,

'Bob': 25,

'Charlie': 35,

};

print('Map of ages: $ages'); // Output: Map of ages: {Alice: 30, Bob: 25, Charlie: 35}

// 5. Runes

// Runes represent Unicode code points. They are useful for handling characters outside the basic multilingual plane.

String heartEmoji = '❤️';

Runes heartRunes = heartEmoji.runes;

print('Runes for heart emoji: ${heartRunes.toList()}'); // Output might be: [10084, 65039]

// 6. Symbols

// Symbols are used to refer to identifiers by name. They are mostly useful in reflection and meta-programming.

Symbol mySymbol = #myExample;

print('Symbol: $mySymbol'); // Output: Symbol("myExample")

// 7. Dynamic

// Dynamic allows the variable’s type to change at runtime, but it sacrifices compile-time type checking.

dynamic variable = 'Hello, world!';

print('Dynamic as String: $variable'); // Output: Dynamic as String: Hello, world!

variable = 123; // Changing type to int

print('Dynamic as int: $variable'); // Output: Dynamic as int: 123

// 8. Object

// Object is the base type for all Dart objects. Every type in Dart is a subclass of Object.

Object genericObject = 'I can be any object';

print('Object: $genericObject'); // Output: Object: I can be any object

}