Maps: Key-Value Pairs

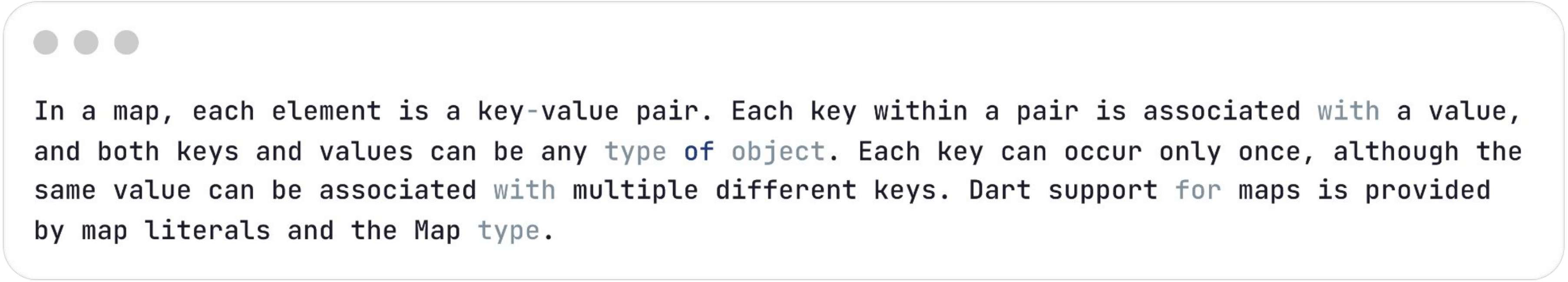
Storing data as key-value associations
Topics: Nested Maps | Methods: keys, values, entries, forEach | Collection if/spread operators

What are Maps?

In programming, **Maps** (also known as dictionaries, hash maps, or associative arrays depending on the language) are **collections that store data in key–value pairs**.

Each **key** is unique and maps to a **specific value**.

- Think of it like a real-world **dictionary**: the *word* is the key, and the *definition* is the value.
- You use the key to find or update the corresponding value quickly.



```
In a map, each element is a key-value pair. Each key within a pair is associated with a value,
and both keys and values can be any type of object. Each key can occur only once, although the
same value can be associated with multiple different keys. Dart support for maps is provided
by map literals and the Map type.
```

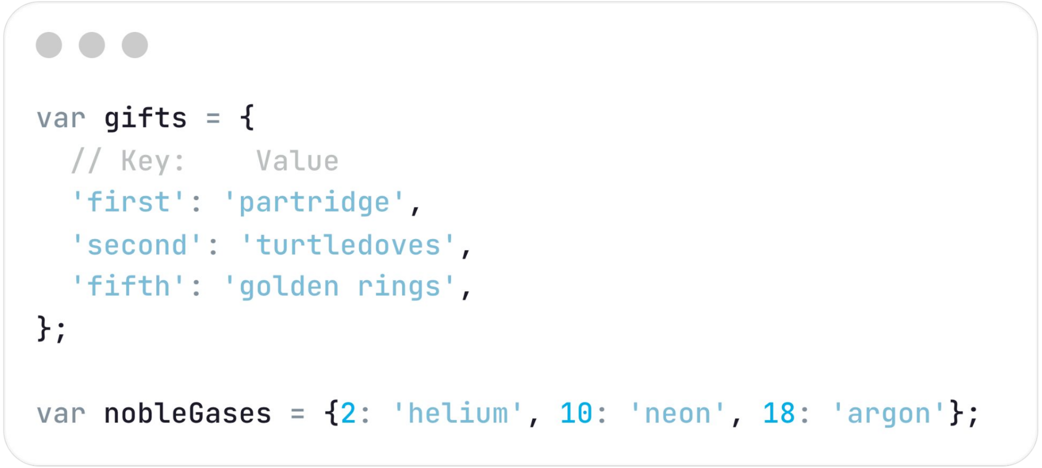
Why Maps Were Needed?

- Early programs relied on **arrays and lists** → could only use integer indexes.
- Real-world data often needs lookup by **names**, **IDs**, or **labels**, not just numbers.
- Linear search through lists was **too slow** for large datasets.
- Need arose for:
  - Fast lookup using flexible keys
  - Better memory usage for sparse data
  - Cleaner abstraction for key-value relationships.

    Motivation: Efficient *key* → *value* retrieval beyond arrays.

Key Characteristics of Map

- Each key is unique.
- Values can be duplicated.
- Fast lookup and insertion.
- Keys can be of many types (e.g., strings, numbers, objects — depending on the language).



```
var gifts = {
  // Key:    Value
  'first': 'partridge',
  'second': 'turtledoves',
  'fifth': 'golden rings',
};

var nobleGases = {2: 'helium', 10: 'neon', 18: 'argon'};
```

Example of Map

```dart
void main() {
  // Creating a Map
  var user = {
    'name': 'Alice',
    'age': 25,
    'country': 'Bangladesh'
  };

  // Accessing values
  print(user['name']);      // Output: Alice

  // Adding or updating values
  user['age'] = 26;

  // Adding a new key-value pair
  user['email'] = 'alice@example.com';

  print(user);
}
```

Common Map Operations — Overview

- Maps let us store and retrieve values efficiently using **keys**.
- Common operations make Maps flexible for everyday use:
    - Lookup
    - Add or Update
    - Check Existence
    - Remove
    - Iterate Keys/Values

Core Operations

- `map[key]` → **Get** value by key
  Example: `user['name'] → "Alice"`
- `map[key] = value` → **Add or update** a key-value pair
  Example: `user['age'] = 26`
- `map.containsKey(key)` → **Check** if a key exists
  Example: `user.containsKey('email')` → `true / false`

Managing and Iterating Data

- `map.remove(key)` → **Remove** a key-value pair
  Example: `user.remove('email')`
- `map.keys` → Access all keys
  Example: `['name', 'age']`
- `map.values` → Access all values
  Example: `['Alice', 26]`
- `map.entries` → Access all entries (key–value pairs)
  Example: `(name: Alice, age: 26)`
- `map.forEach` → Loop through key-value pairs
  Example: `user.forEach((key, value) => …);`

Nested Map

A **"nested map"** refers to a data structure where t**he value associated with a key in a map is itself another map**. This creates a **hierarchical** or **multi-level** organization of data.

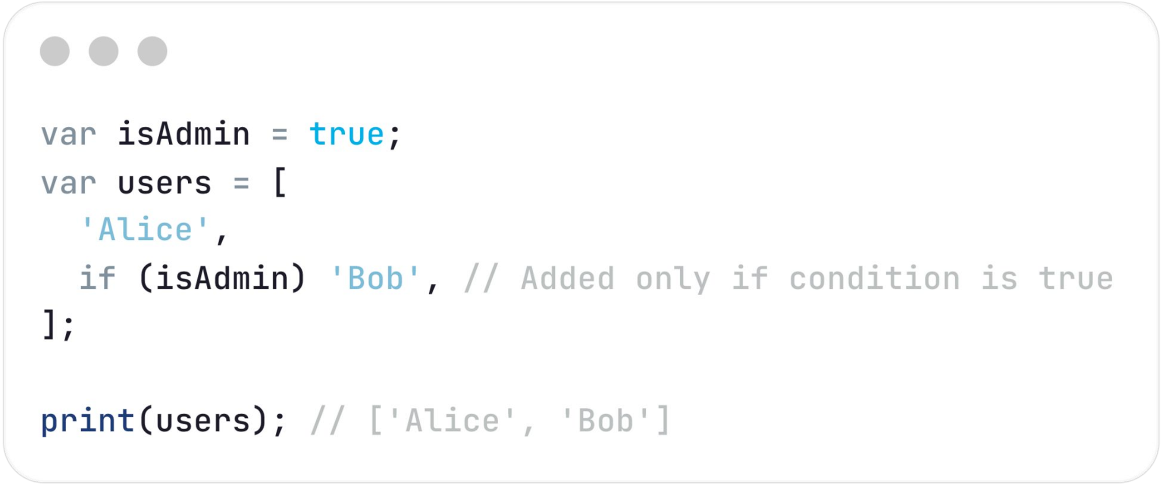
```
var users = {
  'user1': {
    'name': 'Alice',
    'age': 26,
  },
  'user2': {
    'name': 'Bob',
    'age': 30,
  },
};

print(users['user1']?['name']); //Output: Alice
```

Collection If & Spread Operators - 1/2

Collection If

- Allows **conditional elements** inside collections (List, Set, Map).
- Useful for cleaner code without manual if-else wrapping.



```dart
var isAdmin = true;
var users = [
  'Alice',
  if (isAdmin) 'Bob', // Added only if condition is true
];

print(users); // ['Alice', 'Bob']
```

Collection If & Spread Operators - 2/2

Spread Operator (... and ...?)

- Allows inserting multiple elements from another collection.
- ...? handles null collections safely.

```
var base = {'name': 'Alice'};
var extra = {'age': 26};

var user = {
  ...base,
  ...?extra,
};

print(user); // {name: Alice, age: 26}
```

Real World Use Cases of Maps

- **User Profiles / Settings**
  - Store user info like name, age, preferences, roles, etc.
  - Example: `{ 'name': 'Alice', 'theme': 'dark' }`
- **Configuration & Environment Variables**
  - Key-value pairs for system or app settings.
  - Example: `{ 'API_URL': '...', 'MODE': 'production' }`
- **JSON / API Response Handling**
  - Maps are perfect for parsing structured JSON data.
  - Example: API → `Map<String, dynamic>` in Dart.