‹ Operators Metadata ›

Comments

Dart supports single-line comments, multi-line comments, and documentation comments.

Single-line comments

A single-line comment begins with `//`. Everything between `//` and the end of line is ignored by the Dart compiler.

```
void main() {
  // TODO: refactor into an AbstractLlamaGreetingFactory?
  print('Welcome to my Llama farm!');
}
```

Multi-line comments

A multi-line comment begins with `/*` and ends with `*/`. Everything between `/*` and `*/` is ignored by the Dart compiler (unless the comment is a documentation comment; see the next section). Multi-line comments can nest.

```
void main() {
  /*
   * This is a lot of work. Consider raising chickens.

  Llama larry = Llama();
  larry.feed();
  larry.exercise();
  larry.clean();
   */
}
```

Documentation comments

Documentation comments are multi-line or single-line comments that begin with `///` or `/**`. Using `///` on consecutive lines has the same effect as a multi-line doc comment.

Inside a documentation comment, the analyzer ignores all text unless it is enclosed in brackets. Using brackets, you can refer to classes, methods, fields, top-level variables, functions, and parameters. The names in brackets are resolved in the lexical scope of the documented program element.

Here is an example of documentation comments with references to other classes and arguments:

```
/// A domesticated South American camelid (Lama glama).
///
/// Andean cultures have used llamas as meat and pack
/// animals since pre-Hispanic times.
///
/// Just like any other animal, llamas need to eat,
/// so don't forget to [feed] them some [Food].
class Llama {
  String? name;

  /// Feeds your llama [food].
  ///
  /// The typical llama eats one bale of hay per week.
  void feed(Food food) {
    // ...
  }

  /// Exercises your llama with an [activity] for
  /// [timeLimit] minutes.
  void exercise(Activity activity, int timeLimit) {
    // ...
  }
}
```

In the class's generated documentation, `[feed]` becomes a link to the docs for the `feed` method, and `[Food]` becomes a link to the docs for the `Food` class.

To parse Dart code and generate HTML documentation, you can use Dart's documentation generation tool, `dart doc`. For an example of generated documentation, see the Dart API documentation. For advice on how to structure your comments, see Effective Dart: Documentation.

‹ Operators Metadata ›