Dart

Google

Visão geral

- A Dart é uma linguagem desenhada originalmente para a web, que foi concebida na conferência GOTO na Dinamarca em outubro de 2011, em um projeto fundado pelos desenvolvedores Lars Bark e Kasper Lund;

Visão geral

- A Dart é uma linguagem desenhada originalmente para a web, que foi concebida na conferência GOTO na Dinamarca em outubro de 2011, em um projeto fundado pelos desenvolvedores Lars Bark e Kasper Lund;

- Desenhada para facilmente escrever ferramentas de desenvolvimento para aplicações web modernas e capacitadas para ambientes de alta performance;

Visão geral

- A Dart é uma linguagem desenhada originalmente para a web, que foi concebida na conferência GOTO na Dinamarca em outubro de 2011, em um projeto fundado pelos desenvolvedores Lars Bark e Kasper Lund;

- Desenhada para facilmente escrever ferramentas de desenvolvimento para aplicações web modernas e capacitadas para ambientes de alta performance;

- Dart é baseada em classes, não suporta herança múltipla, é orientada a objetos, sendo opcionalmente tipada;

Visão geral

- A Dart é uma linguagem desenhada originalmente para a web, que foi concebida na conferência GOTO na Dinamarca em outubro de 2011, em um projeto fundado pelos desenvolvedores Lars Bark e Kasper Lund;
- Desenhada para facilmente escrever ferramentas de desenvolvimento para aplicações web modernas e capacitadas para ambientes de alta performance;
- Dart é baseada em classes, não suporta herança múltipla, é orientada a objetos, sendo opcionalmente tipada;
- Os programas em Dart podem ser verificados estaticamente, alertando problemas, mas não impedindo a execução;

Visão geral

- A Dart é uma linguagem desenhada originalmente para a web, que foi concebida na conferência GOTO na Dinamarca em outubro de 2011, em um projeto fundado pelos desenvolvedores Lars Bark e Kasper Lund;
- Desenhada para facilmente escrever ferramentas de desenvolvimento para aplicações web modernas e capacitadas para ambientes de alta performance.
- Dart é baseada em classes, não suporta herança múltipla, é orientada a objetos, sendo opcionalmente tipada;
- Os programas em Dart podem ser verificados estaticamente, alertando problemas, mas não impedindo a execução;
- Os programas em Dart podem ser executados no modo de execução e no modo de verificação.

Hello World

Dart

```
void main() {
  print('Hello, World!');
}
```

Variáveis

Dart

```
var name = 'Voyager I';
var year = 1977;
var antennaDiameter = 3.7;
var flybyObjects = ['Jupiter', 'Saturn', 'Uranus', 'Neptune'];
var image = {
  'tags': ['saturn'],
  'url': '//path/to/saturn.jpg'
};
```

Controle de fluxo

Dart

```
if (year >= 2001) {
  print('21st century');
} else if (year >= 1901) {
  print('20th century');
}
for (var object in flybyObjects) {
  print(object);
}
for (int month = 1; month <= 12; month++) {
  print(month);
}
while (year < 2016) {
  year += 1;
}
```

Funções


```dart
int fibonacci(int n) {
  if (n == 0 || n == 1) return n;
  return fibonacci(n - 1) + fibonacci(n - 2);
}
var result = fibonacci(20);
```

Funções


```dart
int fibonacci(int n) {
  if (n == 0 || n == 1) return n;
  return fibonacci(n - 1) + fibonacci(n - 2);
}
var result = fibonacci(20);
```

Imports


```dart
// Importing core libraries
import 'dart:async';
import 'dart:math';

// Importing libraries from external packages
import 'package:angular2/angular2.dart';

// Importing files
import 'path/to/my_other_file.dart';
```

Classes

Dart

```
class Spacecraft {
  String name;
  DateTime launchDate;
  int launchYear;

  // Constructor, including syntactic sugar for assignment to members.
  Spacecraft(this.name, this.launchDate) {
    // Pretend the following is something you'd actually want to run in
    // a constructor.
    launchYear = launchDate?.year;
  }
```

Classes


```
// Named constructor that forwards to the default one.
  Spacecraft.unlaunched(String name) : this(name, null);

  // Method.
  void describe() {
    print('Spacecraft: $name');
    if (launchDate != null) {
      int years = new DateTime.now().difference(launchDate).inDays ~/ 365;
      print('Launched: $launchYear ($years years ago)');
    } else {
      print('Unlaunched');
    }
  }
}
```

Herança

```
class Orbiter extends Spacecraft {
  num altitude;
  Orbiter(String name, DateTime launchDate, this.altitude)
      : super(name, launchDate);
}
```

Interfaces


```
class MockSpaceship implements Spacecraft {
  // ...
}
```

Classes abstratas


```dart
abstract class Describable {
  void describe();

  void describeWithEmphasis() {
    print('=========');
    describe();
    print('=========');
  }
}
```

Async


```
Future<Null> printWithDelay(String message) async {
  await new Future.delayed(const Duration(seconds: 1));
  print(message);
}
```

Async


```
Future<Null> printWithDelay(String message) async {
  await new Future.delayed(const Duration(seconds: 1));
  print(message);
}
```

Exceções

```
if (astronauts == 0) {
  throw new StateError('No astronauts.');
}
```

Exceções

```
if (astronauts == 0) {
  throw new StateError('No astronauts.');
}
```

Getters and setters


```dart
class Spacecraft {
  // ...
  DateTime launchDate;
  int get launchYear => launchDate?.year;
  // ...
}
```