🚀 Cheat Sheet – Dart

Guía rápida de sintaxis y características clave de Dart, incluyendo operadores estándar y especiales.

📦 Variables y Tipos

- `var nombre = "Ana";` – tipo inferido.
- `String apellido = "López";` – tipo explícito.
- `final edad = 25;` – solo lectura (en tiempo de ejecución).
- `const PI = 3.14;` – constante en tiempo de compilación.
- `dynamic x = 5;` – tipo dinámico, menos seguro.
- `late String titulo;` – inicialización diferida.
- `int` – números enteros
- `double` – números decimales
- `bool` – verdadero/falso
- `String` – texto

🔢 Operadores

Aritméticos

- `+` – Suma
- `-` – Resta
- `*` – Multiplicación
- `/` – División (resultado double)
- `~/` – División entera
- `%` – Módulo (resto)
- `-expr` – Negación
- `++var`, `var++` – Incremento
- `--var`, `var--` – Decremento

Comparación

- `==` – Igualdad
- `!=` – Desigualdad
- `>` – Mayor que
- `<` – Menor que
- `>=` – Mayor o igual
- `<=` – Menor o igual

Lógicos

- `!expr` – Negación
- `&&` – AND
- `||` – OR

Asignación

- `=` – Asignación básica
- `+=`, `-=`, `*=`, `/=`, etc.
- `??=` – Asigna solo si es null

Operadores Únicos de Dart

- **Null-aware access** – `?.`
  `usuario?.nombre` – Acceso seguro a nulos
- **Null-coalescing** – `??`
  `print(nombre ?? "Desconocido");` – Valor por defecto
- **Null assertion** – `!`
  `usuario!.nombre` – Asegura que no es null
- **Cascade notation** – `..`
  `persona..nombre = "Ana"..saludar();`

- **Conditional cascade** – `?..`
  `usuario?..actualizarPerfil();`
- **Spread operator** – `...`
  `var lista2 = [0, ...lista1];`
- **Null-aware spread** – `...?`
  `var lista = [1, ...?otraLista];`

🔁 Control de Flujo

Condicionales

- `if (cond) { ... } else if { ... } else { ... }`
- `switch (valor) { case 1: ...; break; default: ... }`
- `cond ? expr1 : expr2` – Operador ternario

Bucles

- `for (var i = 0; i < 5; i++) { ... }`
- `for (var item in lista) { ... }`
- `while (cond) { ... }`
- `do { ... } while (cond);`

📌 Funciones

- Definición: `String saludar(String nombre) { return "Hola $nombre"; }`
- Flecha: `void imprimir() => print("Hola");`
- Parámetros opcionales posicionales: `void saludar([String? nombre])`
- Parámetros nombrados: `void saludar({required String nombre})`
- Valores por defecto: `void saludar({String nombre = "Anónimo"})`
- Funciones anónimas: `var f = (x) => x * 2;`
- Closures: `Function() contador() { int i=0; return () => i++; }`

📚 Colecciones

- Listas: `var lista = [1, 2, 3];`
- Sets: `var conjunto = {1, 2, 3};`
- Mapas: `var mapa = {'a': 1, 'b': 2};`
- Métodos útiles: `map()`, `where()`, `fold()`, `reduce()`
- Collection if: `var l = [1, if (cond) 2];`
- Collection for: `var l = [for (var i in [1,2]) i*2];`

🧬 Programación Orientada a Objetos

- Clases básicas: `class Persona { String nombre; Persona(this.nombre); }`
- Constructores con nombre: `Punto.origin() : x=0, y=0;`
- Factory constructors: `factory Conexion() => _instancia;`
- Getters/Setters: `double get area => ancho * alto;`
- Herencia: `class Perro extends Animal`
- Mixins: `class Pajaro with Volador`
- Interfaces: `class Documento implements Imprimible`
- Clases abstractas: `abstract class Figura { double area(); }`

⏳ Asincronía

- Futures: `Future obtenerDatos() async { ... }`
- Async/await: `var datos = await obtenerDatos();`
- Streams: `await for (var item in stream) { ... }`
- Isolates: `Isolate.spawn(tareaPesada, sendPort);`

🛠️ Herramientas CLI

- Crear proyecto: `dart create nombre_proyecto`
- Ejecutar: `dart run`

- Compilar: `dart compile exe archivo.dart`
- Instalar dependencias: `dart pub get`
- Analizar código: `dart analyze`
- Formatear: `dart format .`
- Generar documentación: `dart doc`
- Ejecutar tests: `dart test`

🧪 Testing

- Pruebas unitarias: `test('suma', () { expect(1+1, 2); });`
- Pruebas async: `test('async', () async { await ... });`
- Pruebas Streams: `expect(stream, emitsInOrder([1,2,3]));`
- Setup/teardown: `setUp(() { ... }); tearDown(() { ... });`

📝 Documentación

- Comentarios doc: `/// Calcula el área`
- Generar docs: `dart doc`
- Ejemplo en docs:

```
/// Ejemplo:
/// ```dart
/// var r = Rectangulo(3,4);
/// ```
```

✅ **Consejo:** Combina operadores estándar con los únicos de Dart (`..` , `??` , etc.) para código más expresivo y seguro contra nulls.