Módulo 2: Fundamentos para el Desarrollo de Aplicaciones Móviles

Lección 4: Declaración de Variables en Dart - var, final y const

Objetivos de la Lección

- Comprender las diferencias entre las palabras clave **`var`**, **`final`** y **`const`** al declarar variables en Dart.
- Explicar cuándo y por qué se debe utilizar cada tipo de declaración.
- Aplicar correctamente la sintaxis de **`var`**, **`final`** y **`const`** para gestionar variables de manera eficiente y segura en Dart.
- Conocer las implicaciones de mutabilidad e inmutabilidad en Dart.

Introducción de la Lección

En **Dart**, existen varias formas de declarar variables, y cada una tiene un propósito específico dependiendo de si necesitas que los valores de las variables sean cambiables (mutables) o constantes (inmutables). Dart ofrece tres palabras clave principales para declarar variables: **`var`**, **`final`**, y **`const`**.

Comprender la diferencia entre estas palabras clave es esencial para gestionar correctamente la memoria, el rendimiento y el comportamiento de los programas. En esta lección, exploraremos las características de cada una y aprenderemos cuándo utilizar cada una para obtener los mejores resultados.

Declaración de Variables en Dart

1. `var`: Variables con Tipado Dinámico

`var` se utiliza para declarar variables cuyo valor puede cambiar más tarde. Aunque Dart es un lenguaje de tipado estático, cuando se usa `var`, el compilador infiere el tipo de dato basado en el valor asignado inicialmente. Esto significa que, una vez que se asigna un valor, no puedes asignar un valor de un tipo diferente a esa variable.

Ejemplo de uso de `var`:

```
void main() {

  var nombre = 'Alice'; // Inferido como String

  var edad = 25;        // Inferido como int



  nombre = 'Bob';       // Cambiar el valor es permitido

  // edad = 'veinticinco'; // Esto causaría un error, ya que
edad es de tipo int

}
```

- **Mutabilidad**: Las variables declaradas con `var` son **mutables**, lo que significa que sus valores pueden cambiar después de la declaración.
- **Tipado Estático**: Aunque no especificas el tipo explícitamente, Dart infiere el tipo en la primera asignación.

2. `final`: Variables de Valor Único (Inmutables)

`final` se utiliza para declarar variables cuyo valor solo puede ser asignado una vez. Una vez que se asigna un valor a una variable `final`, este no puede cambiarse, pero **se puede inicializar en tiempo de ejecución**. Esto significa que puedes asignar un

valor a una variable `final` en el momento en que se ejecute el programa, pero no podrás modificarla después de esa asignación.

Ejemplo de uso de `final`:

```
void main() {
  final nombre = 'Alice'; // Inferido como String
  final edad = 25;        // Inferido como int

  // nombre = 'Bob'; // Error: no se puede cambiar una variable final
}
```

- **Inmutabilidad**: Las variables declaradas con `final` no se pueden modificar después de la primera asignación.
- **Inicialización en Tiempo de Ejecución**: Puedes asignarles un valor que se resuelva en tiempo de ejecución, por ejemplo, usando funciones o cálculos complejos.

3. `const`: Constantes en Tiempo de Compilación

`const` se utiliza para declarar **constantes** en tiempo de compilación. A diferencia de **`final`**, las variables `const` deben ser inicializadas con un valor que esté disponible en **tiempo de compilación**, no en tiempo de ejecución. Esto significa que el valor debe ser conocido y constante en todo momento, y no puede depender de ninguna operación o función dinámica.

Ejemplo de uso de `const`:

```
void main() {
  const pi = 3.1416; // Constante conocida en tiempo de
compilación
  const nombre = 'Dart';

  // pi = 3.14; // Error: no se puede cambiar una constante
}
```

- **Inmutabilidad**: Al igual que `final`, las variables `const` son inmutables, pero además, deben ser conocidas en tiempo de compilación.
- **Uso en Constantes Globales**: `const` se utiliza cuando sabes que el valor nunca cambiará y es fijo durante la vida del programa.

Comparación Entre `var`, `final` y `const`

A continuación, comparamos las principales diferencias entre **`var`**, **`final`** y **`const`**:

Característica	**`var`**	**`final`**	**`const`**
Mutabilidad	Mutable (puede cambiar)	Inmutable (solo se asigna una vez)	Inmutable (constante en tiempo de compilación)
Asignación de tipo	Inferido en la primera asignación	Inferido en la primera asignación	Inferido en la primera asignación
Inicialización	En cualquier momento	Solo una vez, en tiempo de ejecución	Debe estar disponible en tiempo de compilación

Característica	`var`	`final`	`const`
Ejemplo de uso	`var nombre = 'Alice';`	`final nombre = 'Alice';`	`const nombre = 'Alice';`
Uso típico	Variables que pueden cambiar durante la ejecución del programa	Valores que no deben cambiar después de asignarse	Constantes conocidas en tiempo de compilación que no cambiarán

Ejemplos Adicionales y Casos de Uso

Uso de `final` con Funciones Dinámicas:

A diferencia de `const`, las variables `final` pueden ser inicializadas con resultados de funciones o cálculos complejos, siempre que estos se ejecuten antes de la primera asignación.

```
void main() {
  final ahora = DateTime.now(); // Se inicializa en tiempo de ejecución
  print('Fecha actual: $ahora');

  // ahora = DateTime.now(); // Error: no se puede reasignar una variable final
}
```

En este ejemplo, `ahora` toma su valor en tiempo de ejecución, pero no puede modificarse una vez asignado.

Uso de `const` en Listas y Mapas:

Puedes usar `const` para definir listas y mapas inmutables. Los elementos dentro de estas estructuras también serán constantes.

```
void main() {
  const listaConstante = [1, 2, 3];
  const mapaConstante = {'clave1': 'valor1', 'clave2':
'valor2'};

  // listaConstante.add(4); // Error: no se puede modificar una
lista constante
}
```

En este caso, tanto la lista como el mapa son inmutables y no pueden ser modificados después de su declaración.

Buenas Prácticas al Utilizar `var`, `final` y `const`

1. **Usa `var` cuando esperes que el valor de una variable cambie** a lo largo del tiempo, pero sigue confiando en el tipado estático para asegurar la consistencia.
2. **Prefiere `final` para valores que no deberían cambiar** después de ser inicializados, pero que pueden depender de cálculos o funciones dinámicas.
3. **Usa `const` siempre que el valor sea conocido y constante** en tiempo de compilación. Esto es especialmente útil para valores fijos como números matemáticos o configuraciones invariables.
4. **Prioriza `const` sobre `final`** cuando sea posible, ya que las constantes pueden optimizarse mejor en el rendimiento del programa.

Resumen de la Lección

En esta lección, hemos explorado las diferencias clave entre **`var`**, **`final`**, y **`const`** en Dart, entendiendo cuándo y por qué usar cada una para gestionar variables en un programa. Hemos visto que:

- **`var`** se utiliza para declarar variables que pueden cambiar, y su tipo se infiere automáticamente en la primera asignación.
- **`final`** se utiliza para variables que solo deben asignarse una vez y no pueden ser modificadas posteriormente, aunque su valor puede calcularse en tiempo de ejecución.
- **`const`** se utiliza para definir constantes que son conocidas en tiempo de compilación y no cambian nunca.

Cada una tiene su lugar y propósito, y elegir correctamente entre ellas permite escribir código más eficiente, claro y libre de errores.

Actividad de la Lección

Esta actividad te permitirá practicar el uso de `var`, `final` y `const`, ayudando a entender cómo y cuándo utilizar cada una de manera efectiva en diferentes escenarios.

Instrucciones:

1. Declara una variable usando `var` y cambia su valor al menos una vez.
2. Declara una variable `final` que se inicialice con el valor de una función que obtenga la fecha y hora actual.
3. Declara una constante `const` para almacenar el valor de la constante matemática pi (3.1416) y úsala en un cálculo para obtener el área de un círculo.