


Asincronía en Dart

Santiago Vallejo - Pocholo Cataño
Valentina Gonzalez - Duvan Mendez
Esteban Cardona



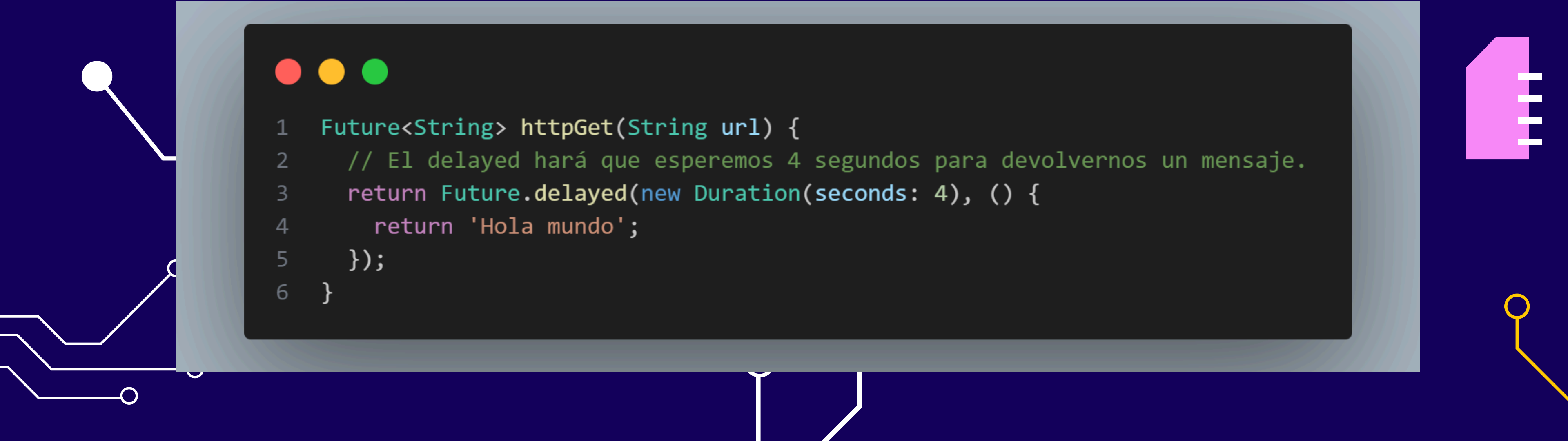
{/} ¿Que es?

Future



Un Future es una tarea asincrónica que se resuelve a destiempo. No es más que un tipo de dato en el que sabemos que se resolverá en un determinado tiempo, no de manera inmediata. Podemos usar un Future para pedir información a un servidor por ejemplo. Sabemos que nos entregará un resultado pero no sabemos cuando.

Así que al ser un tipo de dato podemos hacer lo siguiente. Crear una función que nos devuelva un Future genérico.





```
1 Future<String> httpGet(String url) {  
2     // El delayed hará que esperemos 4 segundos para devolvernos un mensaje.  
3     return Future.delayed(new Duration(seconds: 4), () {  
4         return 'Hola mundo';  
5     });  
6 }
```

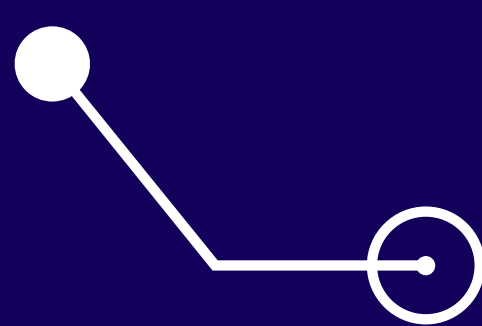


`{/}` ¿Como funciona?

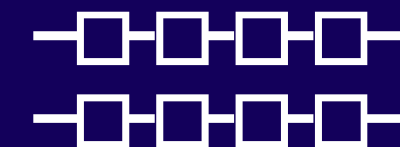
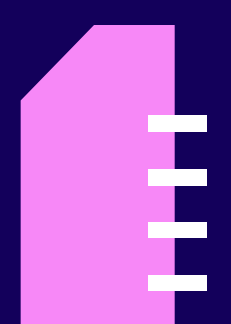
Asincronía



● **async**: La palabra clave async se utiliza para marcar una función como asíncrona. Esto significa que la función puede contener operaciones asíncronas y puede usar await dentro de ella para esperar que esas operaciones se completen sin bloquear el hilo principal de ejecución.

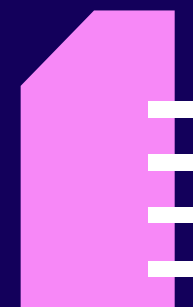


● **await**: La palabra clave await se utiliza dentro de una función marcada con async para esperar la finalización de una operación asíncrona. Cuando se encuentra await, la ejecución de la función se detiene temporalmente hasta que la operación asíncrona se completa y devuelve un resultado. Mientras tanto, el hilo de ejecución puede ocuparse con otras tareas.





```
1  Future<void> obtenerDatos() async {  
2    // Simulando una operación asíncrona que tarda 2 segundos  
3    await Future.delayed(Duration(seconds: 2));  
4    print('Datos obtenidos');  
5  }  
6  
7  void main() async {  
8    print('Iniciando la obtención de datos');  
9    await obtenerDatos(); // Espera a que se completen los datos  
10   print('Datos obtenidos y proceso completado');  
11  }  
12
```



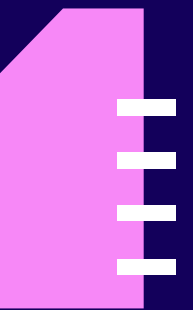
Entonces cómo logramos la asincronía en los datos.

Para este ejemplo usaremos la función hecha anteriormente que nos envía un mensaje con una espera de 4 segundos.

```
1 void main() async {  
2     print('Estamos a punto de pedir datos');  
3     String data = await httpGet('http://codetesthub.com/API/Obtener.php');  
4     print(data);  
5     print('Ultima linea');  
6 }
```


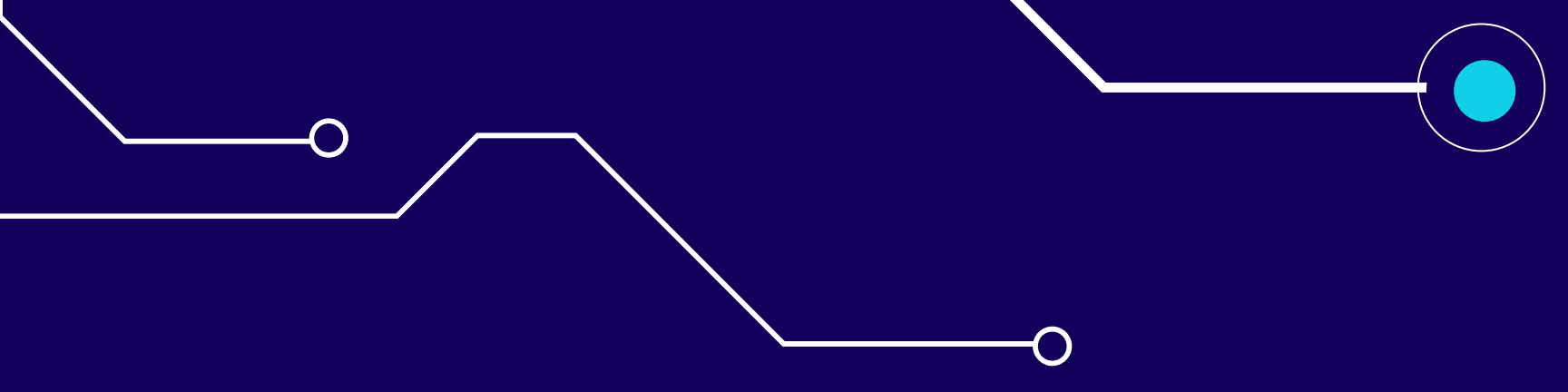


```
1  Future<String> httpGet(String url) {  
2      return Future.delayed(new Duration(seconds: 4), (){  
3          return "Hola mundo";  
4      });  
5  }  
6  
7  void main() async{  
8      print("Estamos a punto de pedir los datos");  
9      String data = await httpGet("http://code.aliens.nasa");  
10     print(data);  
11     print("Ultima linea");  
12 }
```

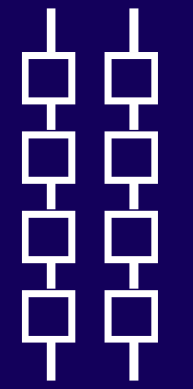


The background is a dark blue field filled with white and light blue geometric patterns. These include thin lines representing circuit traces, small circles of varying sizes, and rectangular blocks. Some elements are solid colors like yellow and cyan, while others are outlines. The overall aesthetic is modern and technological.

Importante



Primero que nada el método debe ser marcado por un `async` para decir al compilador que se trata de una función asíncrona. Entonces usamos el `await` para recibir el `Future<string>` y pedirle que el flujo se espere hasta recibir estos datos. Solo cuando reciba los datos el flujo normal continuará pero solo en esta función marcada con el `async`.



Importante: Para hacer uso del `await` necesitas que la función donde se use necesariamente debe estar marcada con un `async`.

También es importante mencionar que no puedes hacer un constructor asíncrono.





¡Gracias!