

## Dart Class Constructor

2019-04-26 · dart

### 基礎型

先來最簡易的版本，不管從任何語言背景來的人都能看懂。

另外要注意在dart 2.0以後，**new**變成了可選字眼，所以為了程式簡潔，個人建議不要再加**new**。

#### Code

```
1  class Cat{
2    int age;
3    Cat(int age){
4      this.age = age;
5    }
6  }
7  // 使用時
8  var cat = Cat(1);
9  // or 舊版寫法，2.0後亦相容。
10 var cat = new Cat(1);
```

延伸一點，其實參數可以直接使用`this.fieldName`來進行初始化。

可以減少許多冗餘的程式碼。

#### Code

```
1  class Cat{
2    int age;
3    Cat(this.age);
4  }
5  // 使用時
6  var cat = Cat(1);
```

如果想在填參數時能夠直接參照`fieldName`，可以將參數用大括弧括起。

這種做法叫做 **Optional named parameters**，顧名思義，在初始化時甚至可以不給值。

#### Code

```
1  class Cat{
2    int age;
3    Cat({this.age});
4  }
5
6  // 使用時
```



```
7  var cat = Cat(age=1);
8  // or 可以不給值
9  var cat = Cat();
```

另外要注意constructor是不會繼承的，也就是繼承後的物件僅有default constructor。

## Named constructors

Dart是沒有function overloading的，所以若想擁有不同constructor，就需要對這些constructor進行命名。

### Code

```
1  class Cat{
2    int age;
3    Cat.baby(){
4      age = 0;
5    }
6  }
7
8  // 使用時
9  var babyCat = Cat.baby();
```

另外如果想要給class field初始值，還有以下幾種方法

### Code

```
1  class Cat{
2    // 適用於有多個constructor又想指定為同一個初始值時
3    int age = 0;
4    Cat.baby();
5  }
6
7  class Cat{
8    // 適用於有多個constructor但想指定不同初始值時
9    int age;
10   Cat.baby(): age=0;
11   Cat.baby2(): age=0 {
12     //這裡還可以再加寫東西
13   };
14 }
15
16 class Cat{
17   // 適用於Optional named parameters
18   int age;
19   // 新版寫法
20   Cat.baby({this.age=0});
21   // 舊版寫法，可能會被廢棄，不再建議使用。
22   Cat.baby2({this.age:0});
23 }
```



值得注意的是，如果field為final，則第一種(在大括號內初始化)的方法就不再適用，編譯器會報出錯誤訊息。另外這些初始化方法不管是基礎型constructor或是Named constructor皆適用。

## const constructor

這是一個比較難懂的概念，可以先參看下列的範例。

### Code

```
1  class Cat{
2    final int age;
3    const Cat(this.age);
4  }
5
6  // 使用時
7  var a = const Cat(1);
8  var b = const Cat(1);
9  assert(a == b); // true
10
11 // 錯誤使用
12 var c = Cat(1);
13 var d = Cat(1);
14 assert(c == d); // error
```

簡單的說，a跟b為同樣的instance，所以在進行==比較時回傳結果為true。而c跟d為不同的instance，在進行比較時回傳結果為false。

a跟b明明都是由constructor所產生，但卻產生了一樣的instance，因此這個constructor被稱為const constructor。

現在我們來看一下要使用const constructor的規則。

1. class內的所有成員皆為final。
2. 被宣告為const的constructor一定要初始化所有變數。
3. 要使用const constructor時，必須明確的在constructor前再次加上const。(e.g. `var cat = const Cat(1);`)
4. 使用const constructor時，所有傳入的參數必須要是常數(constant)，否則編譯器會報錯。
5. 若使用const constructor時沒加上const前綴，也可以當作一般的constructor用。(e.g. `var cat = Cat(1);`)

因此宣告一個const constructor，並不代表使用這個constructor的結果都為const constructor，這是最大的陷阱。

下面再多給幾個範例。

### Code

```
1  // 可行，傳入變數為constant
2  const a = 1;
3  var cat = const Cat(a);
4
5  // 不可行，final並非constant
```



```
6  final a = 1;
7  var cat = const Cat(a); // error
8
9  // 可行，不加上const前綴則當作一般的constructor
10 final a = 1;
11 var cat = Cat(a);
```

## factory constructor

最後一個要提的constructor是factory constructor，顧名思義，此為factory pattern的語法糖。

### Code

```
1  class Cat{
2    int age;
3    factory Cat.baby(){
4      final cat = Cat();
5      cat.age = 0;
6      return cat;
7    }
8  }
9
10 // 使用時
11 var cat = Cat.baby();
```

這個範例可以很清楚的看到，factory constructor的instance製造是仰賴其他constructor，而且最後還必須return instance。

所以factory constructor根本不是constructor，而是一個factory，只是用個語法糖讓他看起來像constructor罷了。

## Reference

- [官網教學](#)

---

#dart

[← Dart - Mixin](#)

[\[讀書心得\] 圖解Java多線程設計模式 >](#)

