

02

함수



Dart

함수 선언과 호출하기

함수 선언 위치 알아보기

- 다트에서 함수는 톱 레벨과 클래스의 멤버 그리고 다른 함수 내에 선언

• 함수 선언 위치

```
void some1() {  
}  
void some2() {  
    void some3() {  
    }  
    some3();  
}  
class MyClass {  
    void some4() {  
    }  
}
```

함수 선언과 호출하기

함수 선언 위치 알아보기

- 다탈에서는 함수 오버로딩을 제공하지 않습니다.

• 함수 오버로딩 지원 안 함

```
class MyClass {  
    void some() {  
    }  
    void some(int a) { // 함수 이름 중복 오류  
    }  
}
```

함수 선언과 호출하기

매개변수 타입

- 함수의 매개변수는 타입을 명시하거나 var로 선언, 또는 타입을 생략
- 함수의 매개변수를 var로 선언하면 dynamic 타입

• 매개변수 타입에 맞는 데이터로 호출

```
void some1(int? a) {  
}  
  
main() {  
    some1(10);  
    some1(null);  
    some1('hello'); // 오류  
}
```

• var 타입 매개변수

```
void some2(var a) {  
    a = 20;  
    a = 'world';  
    a = true;  
    a = null;  
}  
  
main() {  
    some2(); // 매개변수에 값을 전달하지 않아서 오류  
    some2(10);  
    some2('hello');  
}
```

함수 선언과 호출하기

매개변수 타입

- 매개변수의 타입을 생략하면 var로 선언한 것과 동일

```
• 매개변수 이름 생략 → dynamic 타입

void some3(a) {
  a = 20;
  a = 'world';
  a = true;
  a = null;
}

main() {
  some3(); // 오류
  some3(10);
}
```

함수 선언과 호출하기

함수의 반환 타입

- 반환할 데이터가 없으면 void로 선언
- 반환 타입을 생략하면 모든 타입의 데이터를 반환할 수 있는 dynamic 타입

• 반환 타입

```
void some1() {  
}  
int some2() {  
    return 10;  
}
```

• 반환 타입이 dynamic인 함수에서 return 문 생략

```
dynamic some1() {  
    return 10;  
}  
some2() {  
    return 10;  
}  
some3() {  
}  
  
main() {  
    print('some1 result : ${some1()}');  
    print('some2 result : ${some2()}');  
    print('some3 result : ${some3()}');  
}
```

dynamic 타입

▶ 실행 결과

```
some1 result : 10  
some2 result : 10  
some3 result : null
```

dynamic 타입 함수에 return 문이 없으므로 null 반환

함수 선언과 호출하기

화살표 함수

- 한 줄 함수는 본문을 중괄호로 묶지 않고 화살표 기호(=>)로 나타내는 방법

```
• 화살표 함수 사용

void printUser1() {
    print('hello world');
}

void printUser2() => print('hello world');

main() {
    printUser1();
    printUser2();
}
```

▶ 실행 결과

```
hello world
hello world
```

명명된 매개변수

- 명명된 매개변수(named parameter)
- 옵셔널 위치 매개변수(optional positional parameter)

명명된 매개변수란?

- 명명된 매개변수는 옵셔널이므로 호출할 때 데이터를 전달하지 않을 수도 있으며, 데이터를 전달할 때는 '이름: 값' 형태로 매개변수 이름과 값을 함께 전달

```
void some({String? data1}) {  
    print('data1: $data1');  
}
```

```
some(data1: 'world');  
}
```

The diagram illustrates the named parameter 'data1' in the function call 'some(data1: 'world');'. A bracket connects 'data1' to a box labeled '이름' (name), and another bracket connects the string value 'world' to a box labeled '값' (value).

그림 5-1 명명된 매개변수

명명된 매개변수

명명된 매개변수 선언 규칙

- 명명된 매개변수는 중괄호 {}로 묶어서 선언한다.
- 여러 매개변수를 중괄호로 묶어 명명된 매개변수로 선언할 수 있다.
- 한 함수에서 명명된 매개변수는 한 번만 선언할 수 있으며 순서상 마지막에 선언해야 한다.
- 명명된 매개변수에는 기본값을 설정할 수 있다.

• 명명된 매개변수 선언

```
void some1({String? data2, bool? data3}, int data1) { } // 오류
void some2(int data1, {String? data2, bool? data3}, {int? data4}) { } // 오류
void some3(int data1, {String? data2, bool? data3}) { } // 성공
```

명명된 매개변수

명명된 매개변수 호출 규칙

- 명명된 매개변수에 데이터를 전달하지 않을 수 있다.
- 명명된 매개변수에 데이터를 전달하려면 반드시 이름을 명시해야 한다.
- 명명된 매개변수에 데이터를 전달할 때 선언된 순서와 맞추지 않아도 된다.

• 명명된 매개변수 선언

```
void some(int data1, {String? data2, bool? data3} ) { }
```

• 명명된 매개변수 호출 예

```
❶ some(); // 오류
❷ some(10); // 성공
❸ some(10, 'hello', true); // 오류
❹ some(10, data2: 'hello', data3: true); // 성공
❺ some(10, data3: true, data2: 'hello'); // 성공
❻ some(data2: 'hello', 10, data3: true); // 성공
```

명명된 매개변수

기본 인자 설정하기

- 기본 인자란 함수 호출 때 데이터를 전달받지 못하면 매개변수에 대입하는 기본값

• 기본 인자 설정

```
String myFun({String data = 'hello'}) {  
    return data;  
}  
  
main() {  
    print('myFun() result : ${myFun()}'); // myFun() result : hello  
    print('myFun(world) result : ${myFun(data : "world")}'); // myFun(world) result : world  
}
```

명명된 매개변수

필수 매개변수 선언하기 - required

- 명명된 매개변수에서 required 예약어는 반드시 값을 전달받도록 강제

• 명명된 필수 매개변수 선언

```
someFun({required int arg1}) {  
    print('someFun().. arg1 : $arg1');  
}  
  
main() {  
    someFun();           // 오류  
    someFun(arg1: 10);  // 성공  
}
```

옵셔널 위치 매개변수

- 옵셔널 위치 매개변수로 선언된 함수는 데이터 전달은 자유지만 순서는 맞춰서 호출

```
void some( [ String name = 'hello', int age = 10 ] ) {  
    print('name: $name, age: $age');  
}
```

```
some('world', 20);  
}
```

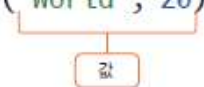


그림 5-2 옵셔널 위치 매개변수

- 매개변수들을 대괄호로 묶는다.
- 함수의 마지막 매개변수에만 사용할 수 있다.
- 매개변수에 기본 인자를 설정할 수 있다.

옵셔널 위치 매개변수

- 옵셔널 위치 매개변수를 포함하는 함수는 다음과 같은 규칙을 지켜 호출
 - 매개변수 이름을 지정할 수 없다.
 - 매개변수가 선언된 순서에 따라 값이 할당된다.

• 옵셔널 위치 매개변수 선언

```
void some(int arg1, [String arg2 = 'hello', bool arg3 = false]) { }
```

• 옵셔널 위치 매개변수 호출

```
❶ some();      // 오류
❷ some(10);    // 성공
❸ some(10, arg2: 'world', arg3: true); // 오류
❹ some(10, 'world', true);           // 성공
❺ some(10, true, 'world');           // 오류
❻ some(10, 'world');                 // 성공
❼ some(10, true);                     // 오류
```

함수 타입 인수

- 함수도 객체
- 함수를 대입할 수 있는 객체를 함수 타입이라고 하며 Function으로 선언

• 함수 타입 선언

```
void some() { }  
Function data2 = some;
```

• 함수를 활용한 예

```
int plus(int no) {  
    return no + 10;  
}  
int multiply(int no) {  
    return no * 10;  
}  
  
Function testFun(Function argFun) {  
    print('argFun : ${argFun(20)}');  
    return multiply;  
}  
  
main(List<String> args) {  
    var result = testFun(plus);  
    print('result : ${result(20)}');  
}
```

▶ 실행 결과

```
argFun : 30  
result : 200
```

함수 타입 인수

- 함수 타입 변수에 대입할 함수를 특정한 형태로 한정하고 싶을 때

• 함수 타입 제한

```
some(int f(int a)) {  
    f(30);  
}  
  
main(List<String> args) {  
    some((int a) {  
        return a + 20;  
    });  
}
```

익명함수

- 이름이 생략된 함수를 의미하며 흔히 람다 함수

• 익명 함수 사용 예

```
fun1(arg) {  
    return 10;  
}  
  
Function fun2 = (arg) {  
    return 10;  
};
```


게터와 세터 함수

- 게터getter와 세터setter 함수는 일반적으로 어떤 데이터를 가져오거나 변경하는 함수를 의미
- get 예약어를 추가한 함수는 데이터를 가져오는 게터
- set 예약어를 추가한 함수는 데이터를 변경하는 세터

• 게터와 세터 선언

```
String _name = 'Hello';

String get name {
    return _name.toUpperCase();
}

set name(value) {
    _name = value;
}
```

• 게터와 세터 호출

```
main(List<String> args) {
    name = "World";
    print('name: $name');
}
```

▶ 실행 결과

name: WORLD

게터와 세터 함수

- get 예약어로 게터만 선언한다면 final 변수처럼 데이터를 가져오기만 할 뿐 바꿀 수는 없습니다.

• 게터만 선언한 예

```
String _name = 'Hello';  
String get name {  
    return _name.toUpperCase();  
}  
  
main(List<String> args) {  
    name = "World"; // 오류  
}
```



감사합니다

단단히 마음먹고 떠난 사람은
산꼭대기에 도착할 수 있다.
산은 올라가는 사람에게만 정복된다.



윌리엄 셰익스피어
William Shakespeare