

---

# Dart 기초 문법

---

# 목차

---

1. 자료형
2. 연산자
3. 조건문, 반복문
4. 함수

- 문자열 변수 사용
- 상수
- null
- enum

```

1 ▼ void main() {
2     //변수 type
3     String name = "Bob";
4     var friend = "Tom"; //type 추론
5     dynamic bestFriend = "Kan"; //type 변경
6
7
8
9     //List
10    List<int> Nums = [1,2,3];
11    Nums[0] = 9; //변경
12
13    //Map key & value pair
14 ▼ Map<String, int> NameAge = {
15        "qwer" : 20,
16        "asdf" : 30
17    };
18    NameAge["qwer"] = 40; //key변경
19 }

```

▶ Run

## 자료형(type)

- int: 정수형
- double: 실수형
- num: int, double을 포함하는 타입
- bool: true, false
- String: 문자열

## - 집합 자료형

- List: 중복을 허용하며 순서가 있는 집합
- Set: 중복을 허용하지 않고 순서가 없는 집합
- Map: key-value 쌍으로 구성된 집합

**dynamic** - 모든 타입을 대변하는 특수 타입!

여러 타입을 한 리스트에 넣거나 일반 변수를 선언할 때도 사용 가능.

List<dynamic> list = [1, 2, 'a']

## 연산자

### 산술 연산자

+: 더하기

-: 빼기

\*: 곱하기

/: 나누기 (double 타입 변환)

~/: 몫(int 타입 변환)

%나머지(int 타입 변환)

### 증감 연산자

전위 연산: ++ [식] --[식]

후위 연산: [식] ++, [식]--

```
void main() {  
    //산술 연산자(+, -, *, /, ~/ , %)  
    var a = 5;  
    var b = 2;  
    print(a ~/ b); // 몫(int)  
    //증감 연산자  
    print(a += 4);  
    //비교, 논리 연산자  
    if (a >= 8 && b == 2) {  
        print("true");  
    }  
}
```

### 비교 연산자

== : 같다

!= : 다르다

> : 더 크다

< : 더 작다

>= : 크거나 같다

<= : 작거나 같다

### 논리 연산자

&&: 그리고

|| : 또는

## 조건문(if), 반복문(for, while)

---

```
void main() {  
    // if-else if-else  
    int score = 90;  
    if (score > 90) {  
        print('A');  
    } else if (score > 80) {  
        print('B');  
    } else {  
        print('C');  
    }  
}
```

```
void main() {  
    // 기본적인 for loop  
    for (int i = 0; i < 10; i++) {  
        print(i);  
    }  
  
    // for-in-loop  
    for (int i in [1, 2, 3, 4, 5]) {  
        print(i);  
    }  
  
    // while loop  
    int i = 10;  
    while (i > 0) {  
        print(i--);  
    }  
  
    // do-while  
    do {  
        print(i--);  
    } while (i > 0);  
}
```

## 함수

---


```
void main() {
    //함수 - 리턴값 타입 함수명 (매개변수 타입 매개변수) {}
    void introduce(String name, [String food = 'chocolate']) {
        //[파라미터] - optional parameter
        //[파라미터 = default] - 기본값 설정
        print('I am $name, I like $food!');
    }
    introduce('Tom');
    introduce('Tom', 'chicken');

    //named parameter - 순서 상관x
    add({
        required int x,
        required int y,
        required int z,
    }) {
        int sum = x+y+z;
        print(sum);
    }
    add(x: 10, y: 20, z: 30);
    add(y: 20, x: 10, z: 30);
}
```

## 문자열 변수 사용

---

```
void main() {  
    void printName(String name) {  
        print("I'm $name.");  
    } // $변수 - 문자열 내에서 어떤 변수의 값을 그대로 사용  
    printName("Bob");  
  
    void printAge(int age) {  
        print("I'm an ${age > 18 ? 'adult' : 'adolescence'}");  
        // 표현식을 써야 할 때는 반드시 ${ }  
        printAge(7);  
    }  
}
```



## 상수(final vs const)

---

```
void main() {  
    const DateTime rightNow = DateTime.now();  
    final DateTime rightNow = DateTime.now(); |  
}
```

현재 시간을 가져오는  
DateTime.now(); 는 런타임  
시점에서 결정되기 때문에  
const에서 에러가 난다

const는 컴퓨터 언어로 번역되는 컴파일 과정에서 상수가  
된다.

>> 런타임 시점에 결정되는 값은 const에 담을 수 없다.

final은 번역 후 실제로 프로그램이 실제 실행되는 과정에서  
상수가 된다.



## nullable / non-nullable

---

```
▼ void main() {  
  
    String food = "chicken";  
    food = null;  
  
    //nullable  
    String? food2 = "hamberger";  
    food2 = null;  
    print(food2);  
  
    //non-nullable  
    String! food3 = "potato";  
    food3 = null;  
    print(food3);  
  
}
```

=> 모든 자료형은 기본적으로 non-nullable

=> ? - nullable

=> ! - non-nullable

## enum

---

```
▼ enum Status {  
    approved,  
    pending,  
    rejected,  
}  
  
▼ void main() {  
    Status status = Status.approved;  
  
    ▼ if(status == Status.approved) {  
        print('승인입니다');  
    }  
    ▼ else if(status == Status.pending) {  
        print('대기입니다');  
    }  
    ▼ else{  
        print('거절입니다');  
    }  
}
```

- 정확히 이값만 존재
- 몇가지 타입만  
사용하도록 강제가능
- 오타 방지