Dart - 함수

상태	플러터

함수 정의

다른 언어랑 같음. 출력방식 지정하고(void면 아무것도 출력하지 않음) 입력 뭐 되는지 지정하고…

```
String sayHello(String name){
 return "Hello $name";
}
```

함수 시작하자마자 바로 리턴할 때는 arrow function으로 쓸 수도 있음

```
num plus(int a, int b) => a+b;
```

Named parameters

일반적으로 여러 개의 매개 변수를 가지는 함수는 아래와 같이 정의하고 사용한다.

```
void sayHello(String name, int age){
  print("Hello $name, you are $age);
}

sayHello("하음", 24)
```

그런데 이렇게 정의하면 순서를 반드시 지켜야하는데, 유저가 순서를 지키지 않을 수도 있다는 문제가 발생한다…

이 때 named parameters를 사용한다.

- 함수를 생성할 때 매개변수에 중괄호를 사용하기
- 함수를 사용할 때 전달인자에 반드시 이름을 적기 : 를 사용한다.

```
void sayHello({String name, int age}){
  print("Hello $name, you are $age);
}

sayHello(name: "하음", age :24)
```

그런데 이러면 한 가지 문제가 발생한다. 유저가 값을 안 주면 어떡할 건데? 하는 문제.

이럴 때에 두 가지 해결방법이 있다.

1. 디폴트 값 만들기

```
void sayHello({String name = "anon", int age = 20}){
  print("Hello $name, you are $age);
}

sayHello(name: "하음", age :24)
```

2. required 사용하기 * 필수값 지정

디폴트 값을 사용하기 싫다면, 유저에게 무조건 값을 입력받아야 한다면 사용한다.

```
void sayHello({
    required String name,
    required int age}){
  print("Hello $name, you are $age);
}

sayHello(name: "하음", age :24)
```

Optional positional parameters

positional parameters를 사용하고 싶은데, 필수 입력값도 아니게 만들고 싶다면?

기본적으로 positional parameters는 모두 required다. 그런데 이 설정을 바꾸고 싶다면 아래와 같이 작성하면 된다.

```
void sayHello({String name, [int? age = 20]}){
  print("Hello $name, you are $age);
}

sayHello(name: "하음")
```

바로바로 디폴트 값을 설정해주고, 대괄호 씌우고, null일 수도 있다고 하기~~

QQ Operator

|| && 이런 operator 당연히 dart에 있으니까 넘어가고 이 강의에서 다룰 것은 ?= 랑 ?? 이래

만약 우리가 이름을 대문자로 바꿔주는 함수를 만든다고 하면…

```
String capitalizeName(String? name){
  if (name != null){
    return name.toUpperCase();
  }
  return 'ANON';
}

void main(){
  capitalizeName('nico');
  capitalizeName(null);
}
```

길게 이렇게 쓸 수 있지만 화살표 함수를 사용하면?

```
String capitalizeName(String? name) => name != null ? name.toUpperCase() : 'ANON';
// 이름이 null이 아니면 대문자로, null이면 'ANON' 리턴

void main(){
  capitalizeName('nico');
  capitalizeName(null);
}
```

위 처럼 쓸 수 있다. 그런데 QQ를 사용할 수도 있다.

```
String capitalizeName(String? name) => name?.toUpperCase() ?? 'ANON';

void main(){
  capitalizeName('nico');
  capitalizeName(null);
}
```

?? 는 좌항이 null이 아니면 좌항을 리턴, 좌항이 null이라면 우항을 리턴한다.

?= 의 사용법은 아래와 같다. 만약 name이 null이라면 우항을 할당하라는 것이다.

```
void main(){
  String? name;
  name ??= 'nico';
}
```

Typedef

자료형이 헷갈릴 때 도움이 될 alias를 만드는 방법. 자료형에 alias를 붙여준다.

큰 파일이나 긴 코드를 다룰 때 유용하다.

예시로 아래 숫자로 된 리스트를 반대로 리턴하는 함수를 보자

```
List<int> reverseListOfNumbers(List<int> list){
  var reversed = list.reversed;
  return reversed.toList();
}
```

List<int>의 설명을 붙이는 방법은 간단하다.

```
typedef ListOfInts = List<int>;

List<int> reverseListOfNumbers(List<int> list){
  var reversed = list.reversed;
  return reversed.toList();
}

// or...
List<int> reverseListOfNumbers(ListOfInts list){
  var reversed = list.reversed;
  return reversed.toList();
}
```

사실 그냥 List<int>임 별명만 바꿔준 것.

구조화 된 데이터의 형태를 지정하고 싶다면 class 사용

어디서 좀 더 많이 쓰는지는 좀 더 공부해보자