Dart 함수 구현 가이드

기본 함수와 고급 함수 활용

학습목표

- Dart의 함수 선언과 구현 방법 이해
- 매개변수와 반환값을 활용한 함수 작성 능력 향상
- 익명 함수(Anonymous Function)의 이해 및 활용
- 비동기 프로그래밍을 위한 Future 활용 방법 습득

3.1 함수 선언하기

- 함수는 특정 작업을 수행하는 코드 블록을 그룹화하는 방법
- void 키워드를 사용하여 반환값이 없는 함수 선언 가능
- main() 함수는 Dart 애플리케이션의 진입점
- DateTime.now()를 활용한 현재 시간 출력 함수 구현

3.2 함수에 매개변수 추가하기

- 함수에 정보를 전달하기 위한 매개변수 선언
- 필수 매개변수와 선택적 매개변수 구분
- 명명된 매개변수는 중괄호 {}로 선언
- 위치 매개변수는 대괄호 []로 선언

3.3 선택적 매개변수 사용하기

- 선택적 매개변수를 사용하면 함수 호출 시 매개변수 생략 가능
- 명명된 매개변수와 위치 매개변수 지원
- 기본값 설정으로 매개변수 생략 시 기본값 사용

3.4 함수에서 값 반환하기

- 함수는 값을 반환할 수 있으며, 반환 타입 명시 필요
- return 키워드를 사용하여 값 반환
- 반환된 값은 변수에 저장하거나 다른 함수에서 사용 가능

3.5 익명 함수 선언하기

- 익명 함수는 이름이 없는 함수
- 함수 리터럴 또는 람다 표현식이라고도 함
- 주로 콜백 함수나 간단한 연산에 사용

3.6 Future를 사용하여 기능 지연 추가하기

- Future는 비동기 연산의 결과를 나타내는 객체
- async 및 await 키워드를 사용하여 비동기 코드 작성
- Future.delayed를 사용하여 특정 시간 동안 지연

getCurrentDateTime.dart

기본적인 시간 출력 함수 구현

getCurrentDateTime.dart

```dart
void main() {
  getCurrentDateTime();
}

void getCurrentDateTime() {
  var timeLondon = DateTime.now();
  print('London: $timeLondon');
}
```

parameterFunction.dart

매개변수를 활용한 시간 계산 함수

parameterFunction.dart

```dart
void main() {
  getCurrentDateTime(-7);
}

void getCurrentDateTime(int hourDifference) {
  var timeNow = DateTime.now();
  var timeDifference = timeNow.add(Duration(hours: hourDifference));

  print('The time now is: $timeNow');
  print('The time minus 7 hours is: $timeDifference');
}
```

optionalParameters.dart

선택적 매개변수를 활용한 인사 함수

optionalParameters.dart

```dart
void main() {
  printGreetingNamed();
  printGreetingNamed(personName: "Rich");
  printGreetingNamed(personName: "Mary", clientId: 001);
}

void printGreetingNamed({
  String personName = 'Stranger',
  int clientId = 999
}) {
  if (personName.contains('Stranger')) {
    print('Employee: $clientId Stranger danger ');
  } else {
    print('Employee: $clientId $personName ');
  }
}
```

returnFunction.dart

값을 반환하는 DateTime 함수

returnFunction.dart

```dart
void main() {
  DateTime timeNow = getCurrentDateTime(0);
  DateTime timeDifference = getCurrentDateTime(-7);

  print('The time now is: $timeNow');
  print('The time minus 7 hours is: $timeDifference');
}

DateTime getCurrentDateTime(int hourDifference) {
  DateTime timeNow = DateTime.now();
  DateTime timeDifference = timeNow.add(Duration(hours: hourDifference));

  return timeDifference;
}
```

anonymousFunction.dart

익명 함수를 활용한 수학 계산

anonymousFunction.dart

```dart
void main() {
  int value = 5;

  int ex1Squared(num1) => num1 * num1;
  int ex1Cubed(num1) => num1 * num1 * num1;

  int ex2Squared(num1) {
    return num1 * num1;
  }

  int ex2Cubed(num1) {
    return num1 * num1 * num1;
  }

  print('EX1: $value squared is ${ex1Squared(value)}');
  print('EX1: $value cubed is ${ex1Cubed(value)}');

  print('EX2: $value squared is ${ex2Squared(value)}');
  print('EX2: $value cubed is ${ex2Cubed(value)}');
}
```

futureFunction.dart

Future를 활용한 비동기 지연 함수

futureFunction.dart

```dart
void main() async {
  int myDelay = 5;

  print('Hello');

  var value = await _customDelay(myDelay);
  var customText = myDelay == 1 ? "second later" : "seconds later";

  print('Its $value $customText');
}

Future<int> _customDelay(int delay) async {
  try {
    await Future.delayed(Duration(seconds: delay));
    return delay;
  } catch (e) {
    print(e);
    return delay;
  }
}
```

요약

- Dart에서 함수는 코드 재사용과 모듈화의 핵심 요소입니다
- 매개변수와 반환값을 활용하여 유연하고 확장 가능한 함수 구현이 가능합니다
- Future를 통한 비동기 프로그래밍으로 효율적인 작업 처리가 가능합니다

END