커플 D-days 앱

핵심 이론

1. StatelessWidget(STL)

- **정의**: `StatelessWidget`은 상태가 없는 위젯으로, UI를 단순히 렌더링하는 데 사용됩니다. 외부에서 전달받은 데이터에 따라 렌더링 결과가 결정되며, 변경이 발생하지 않습니다.
- **특징**:
  - 데이터나 상태가 변경되지 않기 때문에 `build()` 메서드만 호출됩니다.
  - 화면에 고정된 UI 구성 요소(예: 버튼, 이미지, 텍스트 등) 구현에 적합합니다.
  - 앱의 성능에 유리하며, 재구성(rebuild)이 필요 없는 정적인 UI에 주로 사용됩니다.
- **예시**:위 코드에서 `_CoupleImage`는 상태가 변하지 않는 정적인 이미지를 렌더링하는 역할을 합니다.

```
dart
복사편집
class _CoupleImage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Center(
      child: Image.asset(
        'asset/img/middle_image.png',
        height: MediaQuery.of(context).size.height / 2,
      ),
    );
  }
}
```

---

2. StatefulWidget(STF)

- **정의**: `StatefulWidget`은 상태를 관리할 수 있는 위젯으로, 내부 상태 변화에 따라 UI를 동적으로 변경할 수 있습니다.
- **특징**:
  - 상태(state)는 `State` 클래스에서 관리되며, `setState()`를 호출하면 `build()`가 다시 실행되어 UI가 업데이트됩니다.
  - 사용자 입력, 데이터 변경 등 동적인 상호작용이 필요한 경우 사용됩니다.
  - 상태는 위젯 인스턴스와 독립적이며, 위젯이 소멸되거나 재구성될 때 상태를 유지합니다.
- **구조**:
  - `StatefulWidget` 클래스는 UI의 구조를 정의하고,
  - `State` 클래스는 상태와 그 상태가 UI에 어떻게 반영될지를 정의합니다.
- **예시**:위 코드에서 `HomeScreen`은 내부 상태(`firstDay`)를 관리하며, `setState()`를 통해 UI가 동적으로 업데이트됩니다.

```
dart
복사편집
class HomeScreen extends StatefulWidget {
  @override
  State<HomeScreen> createState() => _HomeScreenState();
}

class _HomeScreenState extends State<HomeScreen> {
  DateTime firstDay = DateTime.now();

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Column(
        children: [
          _DDay(
            onHeartPressed: onHeartPressed,
            firstDay: firstDay,
          ),
        ],
      ),
```

```
    );
  }

  void onHeartPressed() {
    setState(() {
      firstDay = firstDay.subtract(Duration(days: 1));
    });
  }
}
```

---

3. CupertinoDatePicker와 Dialog 사용

- **CupertinoDatePicker**:
  - iOS 스타일의 날짜 선택 위젯으로, 날짜/시간을 스크롤 방식으로 선택할 수 있습니다.
  - `onDateTimeChanged` 콜백을 통해 사용자가 선택한 값을 실시간으로 받을 수 있습니다.
  - **장점**: 직관적이고 사용자 친화적인 날짜 선택 인터페이스를 제공합니다.
  - **예시**:

```
dart
복사편집
showCupertinoDialog(
  context: context,
  builder: (BuildContext context) {
    return CupertinoDatePicker(
      mode: CupertinoDatePickerMode.date,
      onDateTimeChanged: (DateTime date) {
        setState(() {
          firstDay = date;
        });
      },
    );
  },
```

```
);
```

- **showCupertinoDialog**:
  - 화면 하단 또는 중앙에 iOS 스타일의 모달을 표시하는 데 사용됩니다.
  - `barrierDismissible: true`를 설정하면 모달 외부 클릭으로 닫을 수 있습니다.

---

4. MediaQuery와 적응형 디자인

- **MediaQuery**:
  - 화면 크기와 관련된 정보를 제공하는 Flutter 도구로, 적응형 UI를 구현하는 데 사용됩니다.
  - `MediaQuery.of(context).size`를 활용해 화면의 너비와 높이를 동적으로 가져올 수 있습니다.
  - **예시**:위 코드는 화면 높이에 따라 이미지 크기를 조정하는 방식으로 적응형 UI를 구현합니다.

```
dart
복사편집
height: MediaQuery.of(context).size.height / 2,
```

---

5. ThemeData와 TextTheme

- **ThemeData**:
  - 앱의 전역 스타일을 설정하는 데 사용됩니다. 글꼴, 색상, 텍스트 스타일 등을 지정할 수 있습니다.
  - 전역 테마를 활용하면 UI의 일관성을 유지하고 유지보수가 쉬워집니다.
- **TextTheme**:
  - 텍스트 스타일을 관리하는 테마로, 앱 내에서 일관된 텍스트 디자인을 제공합니다.
  - **예시**:

```
dart
복사편집
```

```
theme: ThemeData(
  textTheme: TextTheme(
    headlineLarge: TextStyle(
      fontSize: 80.0,
      fontWeight: FontWeight.w700,
      color: Colors.white,
    ),
  ),
);
```

이론 정리

1. **STL과 STF의 차이점**
   - STL은 정적인 UI, STF는 동적인 UI를 구현하는 데 사용되며, 각자의 목적에 맞게 활용하는 것이 중요합니다.
2. **Cupertino 위젯의 장점**
   - iOS 스타일의 UI 구현을 간편하게 할 수 있으며, 직관적인 사용자 경험을 제공합니다.
3. **적응형 UI 구현**
   - `MediaQuery`와 같은 Flutter 도구를 활용하면 다양한 디바이스 크기에서 자연스러운 UI를 구현할 수 있습니다.
4. **날짜 계산의 유용성**
   - `DateTime` 클래스와 `difference` 메서드는 날짜 기반 계산을 간단하게 처리할 수 있도록 도와줍니다.
5. **글로벌 테마 설정**
   - `ThemeData`와 `TextTheme`로 앱의 전체적인 스타일을 관리하며, 일관성을 유지할 수 있음을 배웠습니다.

커플 D-Day 앱의 핵심 코드 정리

핵심 기능 요약

1. **날짜 선택 및 상태 관리**:

- `CupertinoDatePicker`를 통해 사용자가 특정 날짜(`firstDay`)를 선택하면 상태를 업데이트.
- `setState()`를 사용하여 선택한 날짜를 앱 상태에 반영.

2. **D-Day 계산**:
   - 현재 날짜(`DateTime.now()`)와 선택된 날짜(`firstDay`) 간의 차이를 계산하여 D-Day를 표시.
   - 날짜 차이를 계산하는 로직이 앱의 핵심 기능.
3. **동적 UI 업데이트**:
   - 날짜가 변경될 때마다 상태를 기반으로 D-Day 텍스트와 UI를 동적으로 업데이트.

핵심 코드

1. **날짜 선택 로직**:

```
dart
복사편집
void onHeartPressed() {
  showCupertinoDialog(
    context: context,
    builder: (BuildContext context) {
      return Align(
        alignment: Alignment.bottomCenter,
        child: Container(
          color: Colors.white,
          height: 300,
          child: CupertinoDatePicker(
            mode: CupertinoDatePickerMode.date,
            onDateTimeChanged: (DateTime date) {
              setState(() {
                firstDay = date; // 선택된 날짜를 상태로 저장
              });
            },
          ),
        ),
      );
    },
```

```
    barrierDismissible: true, // 외부 클릭으로 닫기 허용
  );
}
```

2. **D-Day 계산 로직**:

```
dart
복사편집
final int dDay = DateTime.now().isAfter(firstDay)
    ? DateTime.now().difference(firstDay).inDays
    : firstDay.difference(DateTime.now()).inDays;
```

3. **D-Day 표시 로직**:

```
dart
복사편집
Text(
  'D+$dDay', // D-Day 결과를 동적으로 표시
  style: Theme.of(context).textTheme.headlineMedium,
),
```

핵심 구현의 중요성

1. **날짜 선택과 상태 변경**:
   - `onHeartPressed` 함수는 사용자의 입력을 받아 상태(`firstDay`)를 동적으로 업데이트하는 앱의 중요한 인터페이스 역할을 함.
2. **날짜 차이 계산**:
   - `DateTime.difference` 메서드는 두 날짜 간의 차이를 간단히 계산할 수 있는 강력한 도구로, 앱의 D-Day 계산 핵심 로직을 담당.
3. **UI와 상태의 연결**:
   - Flutter의 상태 관리(`setState`)와 빌드 함수가 연동되어, 상태 변화가 UI에 즉각적으로 반영되는 구조를 보여줌.


Android Emulator - Medium_Phone_API_35:5554
8:20
DEBUG
SSAFY
우리 처음 만난 날
2024.4.20
D+272