안드로이드 무효화란 무엇인가?

안드로이드 View 시스템의 무효화(Invalidation) 매커니즘

다이스

무효화(Invalidation)란?

무효화는 View를 다시 그려야 함을 시스템에 알리는 과정입니다

무효화(Invalidation)란?

무효화는 View를 다시 그려야 함을 시스템에 알리는 과정입니다

버튼의 색상이나 위치가 변경되었다면, 사용자가 실제로 변경된 상태를 볼 수 있도록 화면을 다시 그려야 합니다

이때 시스템은 무효화를 통해 어떤 View를 다시 그려야 하는지 알게 됩니다

무효화(Invalidation)란?

무효화는 View를 다시 그려야 함을 시스템에 알리는 과정입니다

버튼의 색상이나 위치가 변경되었다면, 사용자가 실제로 변경된 상태를 볼 수 있도록 화면을 다시 그려야 합니다

이때 시스템은 무효화를 통해 어떤 View를 다시 그려야 하는지 알게 됩니다

왜 필요한가?

- UI 변경사항(버튼 색상, 위치 등)이 발생했을 때 화면을 다시 그리도록 요청
- 모든 View를 매 프레임마다 다시 그리는 것은 성능 저하를 초래
- 필요한 순간에만 UI를 갱신하여 성능 최적화

View 그려지는 과정

안드로이드 View는 화면에 표시될 때

측정(Measure) → 레이아웃(Layout) → 그리기(Draw) 과정으로 그려집니다

Measure

View의 크기와 자식 View의 크기를 결정

Layout

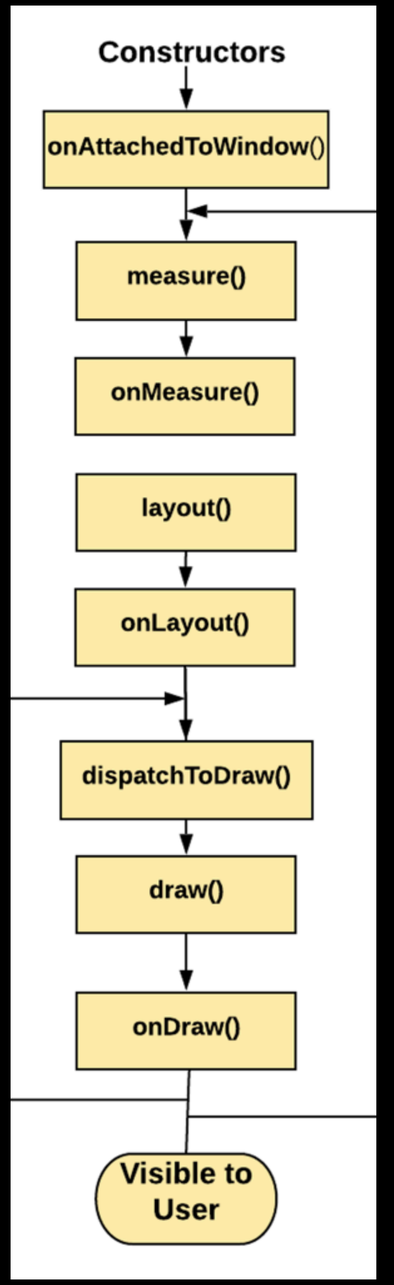
View와 자식 View의 위치를 결정

Draw

실제 화면에 View를 그림

View 그려지는 과정

Android 프레임워크는 측정 패스와 레이아웃 패스라는 두 패스로 레이아웃을 그립니다



View 그려지는 과정

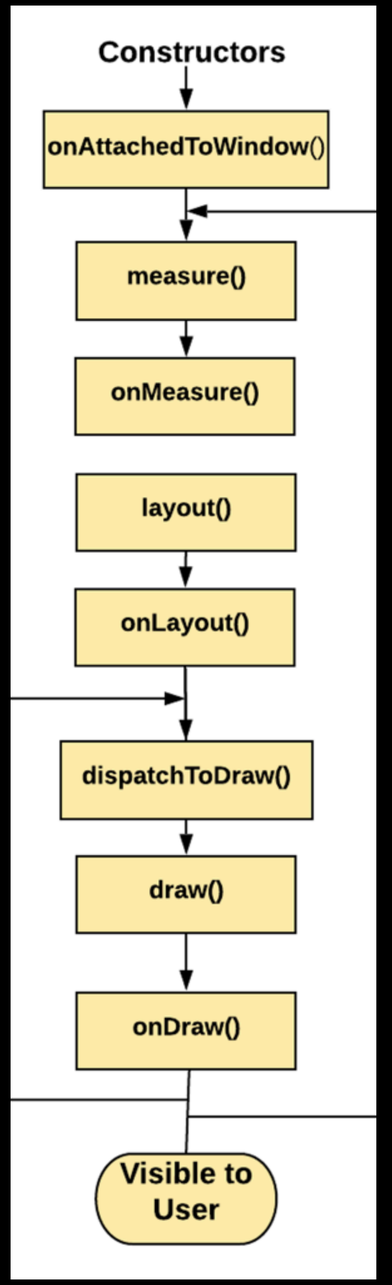
Android 프레임워크는 측정 패스와 레이아웃 패스라는 두 패스로 레이아웃을 그립니다

측정 패스는 View의 크기를 측정하는 단계

Measure 단계에서 ViewGroup.LayoutParams를 사용하여 View의 크기를 알아보고

MeasureSpec을 생성하여 onMeasure를 호출한 후

onMeasure에서 MeasureSpec에 기반한 최종적인 View의 크기를 결정



View 그려지는 과정

Android 프레임워크는 측정 패스와 레이아웃 패스라는 두 패스로 레이아웃을 그립니다

측정 패스는 View의 크기를 측정하는 단계

Measure 단계에서 ViewGroup.LayoutParams를 사용하여 View의 크기를 알아보고

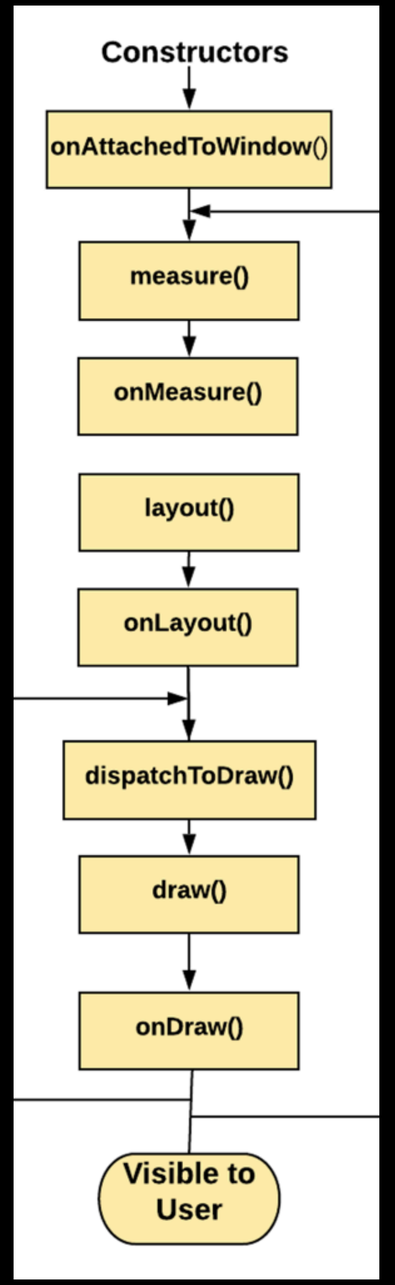
MeasureSpec을 생성하여 onMeasure를 호출한 후

onMeasure에서 MeasureSpec에 기반한 최종적인 View의 크기를 결정

레이아웃 패스는 View를 배치하는 단계

측정 패스에서 계산된 크기를 사용하여 모든 하위 요소를 배치

Layout 메서드로 호출이 시작되고, 이는 뷰의 최종적인 위치를 지정함



View 그려지는 과정

안드로이드 View는 화면에 표시될 때

측정(Measure) → 레이아웃(Layout) → 그리기(Draw) 과정으로 그려집니다

Measure

View의 크기와 자식 View의 크기를 결정

Layout

View와 자식 View의 위치를 결정

Draw

실제 화면에 View를 그림

invalide() 호출 시 View는 다시 그려지지만 Measure와 Layout 단계를 건너뜁니다

즉, 무효화는 Draw 단계와 직접적으로 연결됩니다

무효화 프로세스

1. View에서 invalidate() 또는 postInvalidate()를 호출
2. 해당 View가 dirty 상태로 표시됨
3. 다음 드로잉 프레임에서 ViewRootImpl이 dirty로 표시된 View를 감지
4. draw() 호출 → onDraw() 메서드 실행
5. 화면 갱신

무효화 관련 주요 메서드

invalidate()

단일 View를 무효화하여 다음 프레임에서 다시 그리도록 예약합니다

- UI 스레드에서만 사용 가능
- View의 일부나 전체 내용을 변경할 때 호출
- 해당 View를 ‘dirty’ 상태로 표시하고 다음 레이아웃/드로잉 패스에서 다시 그리도록 예약

```
myView.setColor(Color.BLUE);
myView.invalidate();
```

invalidate(Rect dirty)

View 내 특정 영역만 무효화하여 부분적으로 다시 그립니다

- 전체가 아닌 일부 영역만 업데이트할 때 성능 최적화 가능
- 변경된 영역만 지정하여 불필요한 그리기 작업 방지

```
val dirtyRegion: Rect = new Rect(10, 10, 50, 50)
myView.invalidate(dirtyRegion)
```

postInvalidate()

백그라운드 스레드에서 안전하게 View 무효화 요청을 메인 스레드로 전달합니다

- UI 스레드가 아닌 곳 (네트워크, 데이터 로딩 스레드)에서 사용
- 무효화 요청을 메인 스레드에 게시하여 스레드 안정성 보장
- 비동기 작업 종료 후 UI 갱신 시 필수적으로 사용

```
Thread {
    myView.postInvalidate()
}.start()
```

무효화 관련 주요 메서드

무효화 메서드들은 즉시 View를 다시 그리지 않고,
다음 UI 렌더링 사이클에서 View가 다시 그려지도록 예약하는 역할을 합니다

효율적인 UI 갱신을 위해 상황에 맞는 메서드를 선택할 필요가 있다

무효화 사용 예시

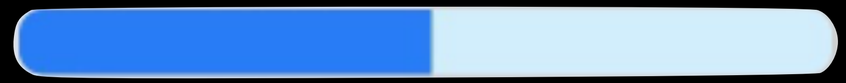
무효화 사용 예시

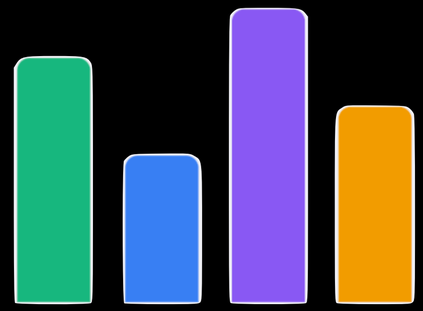
Custom View, 애니메이션, 동적 UI 등 실시간으로 변경되는 요소에 주로 사용합니다

ProgressBar 실시간 업데이트

애니메이션 구현

데이터 시각화





Custom View가 아닌 일반 UI에서는 거의 사용되지 않습니다

진행률 표시, 차트, 그래프, 캔버스 기반 UI 등

실시간으로 동적 변경이 필요한 View나 애니메이션을 직접 구현할 때 사용됩니다

무효화 어떻게 사용은 어떻게?

부분 업데이트 활용

- 전체 View가 아닌 변경된 영역만 다시 그리기
- invalidate(Rect dirty) 메서드 사용
- 불필요한 렌더링 과정 절약으로 성능 향상

백그라운드 스레드

- UI 스레드가 아닌 곳에서는 반드시 `postInvalidate()` 사용
- 네트워크 응답이나 백그라운드 작업 후 UI 갱신 시 적용
- 스레드 안정성 보장

주의사항

- **불필요한 호출 자제**

  → 애니메이션이나 복잡한 레이아웃에서 과도한 invalidate() 호출은 성능 병목 발생의 원인

- 중첩된 View 구조에서 상위 View의 무효화는 하위 요소들도 영향을 받음

무효화는 View 체계에서 UI 변경을 효율적이고 안전하게 반영하는 핵심 매커니즘

효율적인 무효화는 앱 성능과 사용자 경험을 결정하는 핵심 요소