# objectVal 와 onValue 똑 같이 실시간 데이터를 가져다 주는데 어떤 차이가 있는지 ?

네, $\text{Firebase Realtime Database}$ (RTDB)에서 $\text{objectVal}$과 $\text{onValue}$는 모두 실시간 데이터를 가져오는 데 사용되지만, **어떤 방식으로 데이터를 가져오고** $\text{`onValue`}$ 내에서 **어떻게 데이터에 접근하는지**에 차이가 있습니다.

두 함수는 사용 환경이 다릅니다:

1. **onValue**: $\text{Firebase Realtime Database SDK}$의 **표준 메서드**입니다.
2. **objectVal**: 보통 **AngularFire** (특히 레거시 버전의 $\text{AngularFire2}$ 또는 특정 유틸리티)에서 데이터를 객체 형태로 쉽게 변환하기 위해 사용되던 **구조화된 데이터 형태** 또는 **헬퍼 함수**로 언급됩니다. $\text{Firebase SDK}$ 자체에는 $\text{objectVal}$이라는 메서드가 없습니다.

**🔎 $\text{onValue}$ (표준 Realtime Database SDK)**

onValue는 $\text{Firebase Realtime Database SDK}$에서 데이터를 **실시간으로 구독**하는 기본적인 방법입니다.

* **용도**: 특정 $\text{Database}$ 레퍼런스(ref)의 데이터 변경을 **수신 대기**합니다. 데이터가 생성, 업데이트 또는 삭제될 때마다 콜백 함수가 실행됩니다.
* **반환 값**: 콜백 함수는 항상 **$\text{DataSnapshot}$** 객체를 인수로 받습니다.
* **데이터 접근**: $\text{DataSnapshot}$ 객체에서 실제 $\text{JSON}$ 데이터에 접근하려면 반드시 **.val()** 메서드를 사용해야 합니다.

**예시 (JavaScript/TypeScript):**

TypeScript

import { ref, onValue } from "firebase/database";

// 콜백 함수가 DataSnapshot 객체를 받음

onValue(ref(db, 'users/123'), (snapshot) => {

// 실제 데이터에 접근하려면 .val()을 사용해야 합니다.

const userData = snapshot.val();

console.log(userData);

// 키가 존재하는지, 데이터가 있는지 확인하려면 .exists() 사용

const exists = snapshot.exists();

console.log(exists);

});

**💡 $\text{objectVal}$ (AngularFire 또는 유틸리티 패턴)**

objectVal은 $\text{Firebase SDK}$의 메서드가 아니라, 주로 **데이터를 구조화하는 방식**을 설명하거나 $\text{AngularFire}$와 같은 라이브러리가 $\text{RTDB}$의 $\text{`onValue`}$ 스트림을 **가공하여 객체 형태로 제공**할 때 사용되는 **개념**입니다.

**1. $\text{AngularFire}$에서의 'Object' 구독**

$\text{AngularFire}$는 $\text{RTDB}$의 데이터를 $\text{RxJS}$의 $\text{Observable}$ 스트림으로 변환하여 제공합니다. 이 스트림을 구독할 때 데이터는 이미 $\text{JSON}$ 객체로 변환되어 전달됩니다.

* **AngularFireDatabase.object('path')**: 이 메서드를 사용하여 구독하면, 스트림을 통해 전달받는 데이터가 이미 $\text{DataSnapshot.val()}$의 결과인 **표준 JavaScript 객체 형태**로 나옵니다.
  + **차이점**: $\text{onValue}$처럼 .val()을 호출할 필요가 없습니다. 라이브러리가 이 변환을 대신 해줍니다.

**2. 콜백 함수 내의 objectVal (가정)**

만약 $\text{`onValue`}$ 콜백 함수 내에서 $\text{objectVal}$이라는 변수를 사용한다면, 그것은 개발자가 다음과 같이 $\text{snapshot.val()}$의 결과를 담기 위해 **직접 정의한 변수 이름일 가능성**이 높습니다.

TypeScript

onValue(ref(db, 'items'), (snapshot) => {

// 개발자가snapshot.val()의 결과를 담기 위해 objectVal이라는 변수명을 사용

const objectVal = snapshot.val();

console.log(objectVal); // 객체 형태의 데이터

});

**➡️ 핵심 요약**

| **특징** | **onValue (표준 RTDB SDK)** | **objectVal (AngularFire 또는 패턴)** |
| --- | --- | --- |
| **역할** | 데이터 **스트림을 구독**하는 메서드 | **구독된 데이터의 형태** (일반 $\text{JSON}$ 객체) |
| **반환값** | 콜백 함수가 **DataSnapshot** 객체를 받음 | Observable 또는 콜백을 통해 **표준 $\text{JavaScript}$ 객체**를 받음 |
| **데이터 접근** | **.val()** 메서드를 명시적으로 호출해야 함 | 이미 객체로 변환되어 있어 **.val() 호출 불필요** |
| **사용 환경** | $\text{Vanilla Firebase SDK}$ | $\text{AngularFire}$ 또는 사용자 정의 헬퍼 함수 내 |

결론적으로, \*\*onValue는 구독하는 '행위'\*\*이고, \*\*objectVal은 구독을 통해 얻게 되는 '데이터 형태'\*\*를 지칭한다고 이해하시면 됩니다.