In Angular 19, Signals are a key feature for managing state and reactivity within your components.1 They offer a more declarative and performant way to handle data changes compared to traditional approaches like observables or change detection.2

**Key Concepts:**

* **Signals:**
  + A wrapper around a value that notifies consumers (like templates or other signals) when that value changes.3
  + Created using the signal() function.
  + Can hold any type of data.4
* **Writable Signals:**
  + Can be updated directly using the .set() method.
* **Computed Signals:**
  + Derived from other signals using a function.5
  + Automatically update whenever the values of the signals they depend on change.6
* **Effects:**
  + Side effects that are triggered when a signal changes.7
  + Useful for performing actions like updating the DOM, making API calls, or subscribing to other observables.

**Example:**

TypeScript

import { Component, signal } from '@angular/core';

@Component({

selector: 'app-my-component',

template: `

<p>Count: {{ count() }}</p>

<button (click)="increment()">Increment</button>

`,

})

export class MyComponent {

count = signal(0);

increment() {

this.count.set(this.count() + 1);

}

}

In this example:

* count is a signal that holds the current count value.
* The template displays the value of the count signal.
* Clicking the button calls the increment() method, which updates the count signal.
* The template automatically re-renders to reflect the updated count value.8

**Benefits of Using Signals:**

* **Improved Performance:** Signals can lead to more efficient change detection, especially in components with complex state.9
* **Declarative Approach:** Make your component logic more readable and easier to understand.10
* **Better Control:** Fine-grained control over how and when data changes trigger updates.11

**In Angular 19:**

* Signals are further enhanced with features like linkedSignal for managing local state that depends on other signals.
* The Signals API is continuously evolving to provide a more robust and powerful way to manage state in Angular applications.12

By effectively utilizing Signals, you can build more performant, maintainable, and reactive Angular applications.13