# Angular Signals Demo

## 1. Setup Project

### 1.1 Install Dependencies

1. Change directory to calab:

* cd calab

1. Install dependencies by running the following command:

* npm install

## 2. Creating a Signal

### 2.1 Create new Signal

1. Open src/app/app.component.ts file and do the following:
   * Import signal form '@angular/core'.
   * Inside constructor of AppComponent create a new signal and assign it to a variable:
   * const quantity = signal(0);

### 2.2 Read value from the Signal

1. Open src/app/app.component.ts file and do the following:
   * Import effect form '@angular/core'.
   * Inside constructor of AppComponent, just below Signal, call an effect and log the Signal value:
   * effect(() => {  
      console.log(`The current quantity is: ${quantity()}`);  
     });

### 2.3 Update Signal value

1. Open src/app/app.component.ts file and do the following:
   * Inside constructor of AppComponent, just below the effect, set new value to the Signal:
   * quantity.set(1);
   * Now use alternative approach to change value in Signal called update:
   * quantity.update(value => value + 1);

### 2.4 Review Changes

1. Start Angular Development Server if not yet started:

* npx -p @angular/cli ng serve
* *Otherwise refresh the browser tab to see updated view.*

1. Inspect developer console if using Chrome for any logs. You should see the following:

* The current quantity is: 2

## 3. toSignal Example

### 3.1 Create an Observable

1. Open src/app/app.component.ts file and do the following:
   * Import Observable form rxjs.
   * Replace code inside constructor of AppComponentto the following:
   * let increasingQuantity = new Observable<number>(observer => {  
      let curVal = 0;  
      setInterval(() => observer.next(curVal +=1), 5000);  
     });

### 3.2 Convert Observable to Signal

1. Continue working with src/app/app.component.ts file:
   * Import toSignal form @angular/core/rxjs-interop.
   * Just below previously declared Observable, add the following:
   * let increasingQuantitySignal = toSignal(increasingQuantity, {initialValue: 0});

### 3.3 Read value from the newly converted Signal

1. Continue working with src/app/app.component.ts file:
   * Import effect form '@angular/core'.
   * Inside constructor of AppComponent, just below Signal, call an effect and log the Signal value:
   * effect(() => {  
      console.log(`The current quantity is: ${increasingQuantitySignal()}`);  
     });

### 3.4 Review Changes

1. Start Angular Development Server if not yet started:

* npx -p @angular/cli ng serve
* *Otherwise refresh the browser tab to see updated view.*

1. Inspect developer console if using Chrome for any logs. You should see the following geting printed every 5 seconds:

* The current quantity is: 1  
  .  
  .  
  .  
  The current quantity is: 2  
  .  
  .  
  .

## 4. toObservable Example

### 4.1 Create a Signal

1. Open src/app/app.component.ts file and do the following:
   * Import signal form @angular/core.
   * Replace code inside constructor of AppComponentto the following:
   * let quantity = signal(0);

### 4.2 Convert Signal to Observable

1. Continue working with src/app/app.component.ts file:
   * Import toObservable form @angular/core/rxjs-interop.
   * Just below previously declared Signal, add the following:
   * let quantityObservable = toObservable(quantity);

### 4.3 Subscribe to observable to get async updates

1. Continue working with src/app/app.component.ts file:
   * Just below previously converted signal into observable add the following:
   * quantityObservable.subscribe({  
      next: (v) => console.log(`Subscriber: The current quantity is: ${v}`),  
      error: (e) => console.error(e),  
      complete: () => console.info('complete')   
     });

### 4.4 Review Changes

1. Start Angular Development Server if not yet started:

* npx -p @angular/cli ng serve
* *Otherwise refresh the browser tab to see updated view.*

1. Inspect developer console if using Chrome for any logs. You should see the following geting printed:

* Subscriber: The current quantity is: 0  
  complete