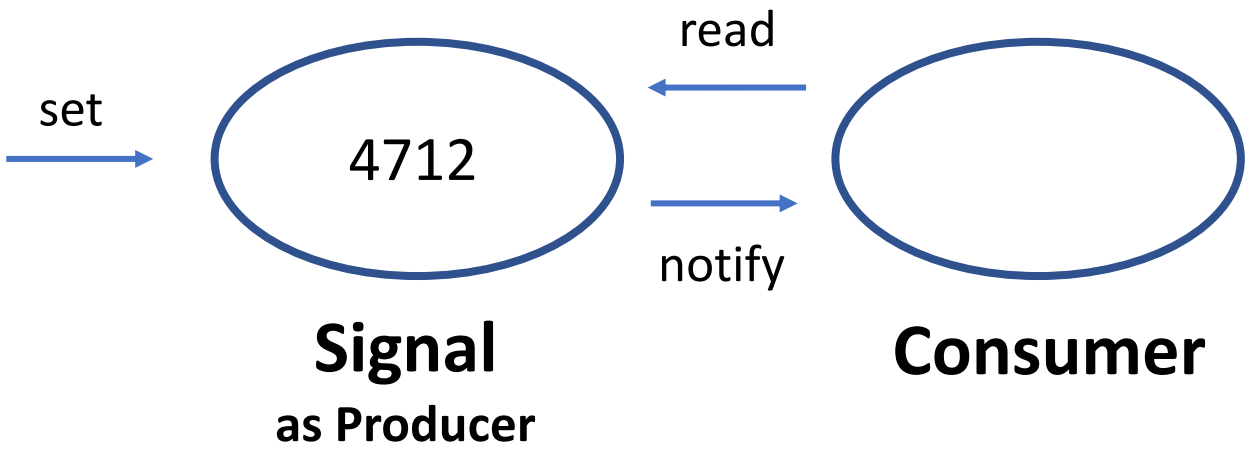

ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE
State Management with Signals and the
NGRX Signal Store
ManfredSteyer

Signals: Simple Reactivity



Component With Signals

```
flights = signal<Flight[]>([]);
```

```
const flights = await this.flightService.findAsPromise(from, to);
this.flights.set(flights);
```

```
<div *ngFor="let f of flights()">
    <flight-card [item]="f" />
</div>
```

Benefits

Simple Reactive Building Block

Fine-grained CD

Zone-less CD

Interop with RxJS

No need to unsubscribe

How will Signals influence State Management?

ANGULAR ARCHITECTS
INSIDE KNOWLEDGE

Agenda

#1
Signals 101

#2
Services +
Signals

#3
NGRX +
Signals

#4
Upcoming: NGRX
Signal Store

#1:
Signals 101

ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

DEMO

(branch: signals)

ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

#2:
Services + Signals

ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

A First Solution

```
@Injectable({ providedIn: 'root' })
export class FlightBookingFacade {
  […]

  private _flights = signal<Flight[]>([]);
  readonly flights = this._flights.asReadonly();

  async load(from: string, to: string) {
    const flights = await […];
    this._flights.set(flights);
  }
}
```

A First Solution

```
@Injectable({ providedIn: 'root' })
export class FlightBookingFacade {
  […]

  private _flights = signal<Flight[]>([]);
  readonly flights = this._flights.asReadonly();

  async load(from: string, to: string) {
    const flights = await […];
    this._flights.set(flights);
  }
}
```

A Bit Verbose …

```
@Injectable({ providedIn: 'root' })
export class FlightBookingFacade {
  […]
  private _flights = signal<Flight[]>([]);
  readonly flights = this._flights.asReadonly();

  private _from = signal('Hamburg');
  readonly from = this._from.asReadonly();

  private _to = signal('Graz');
  readonly to = this._to.asReadonly();
  […]
}
```

State Signal

```
@Injectable({ providedIn: 'root' })
export class FlightBookingFacade {
  […]
  private state = signal({
    from: 'Hamburg',
    to: 'Graz',
    flights: [] as Flight[], […]
  }, { equal });

  […]
}
```

State Signal

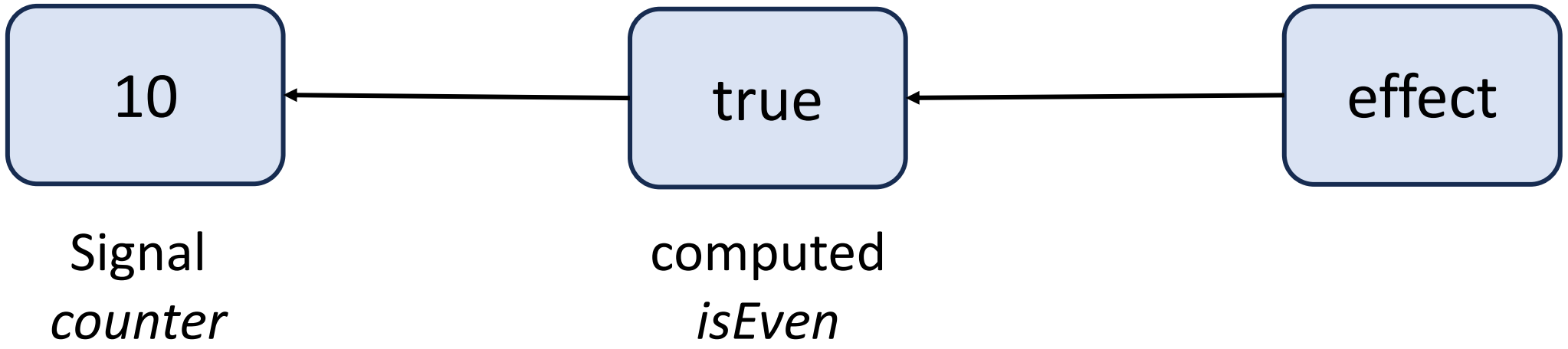
```
@Injectable({ providedIn: 'root' })
export class FlightBookingFacade {
  […]
  private state = signal({
    from: 'Hamburg',
    to: 'Graz',
    flights: [] as Flight[], […]
  }, { equal });

  flights = computed(() => this.state().flights, { equal });
  from = computed(() => this.state().from, { equal });
  […]
}
```

State Signal

```
const equal = (a,b) => a === b
```

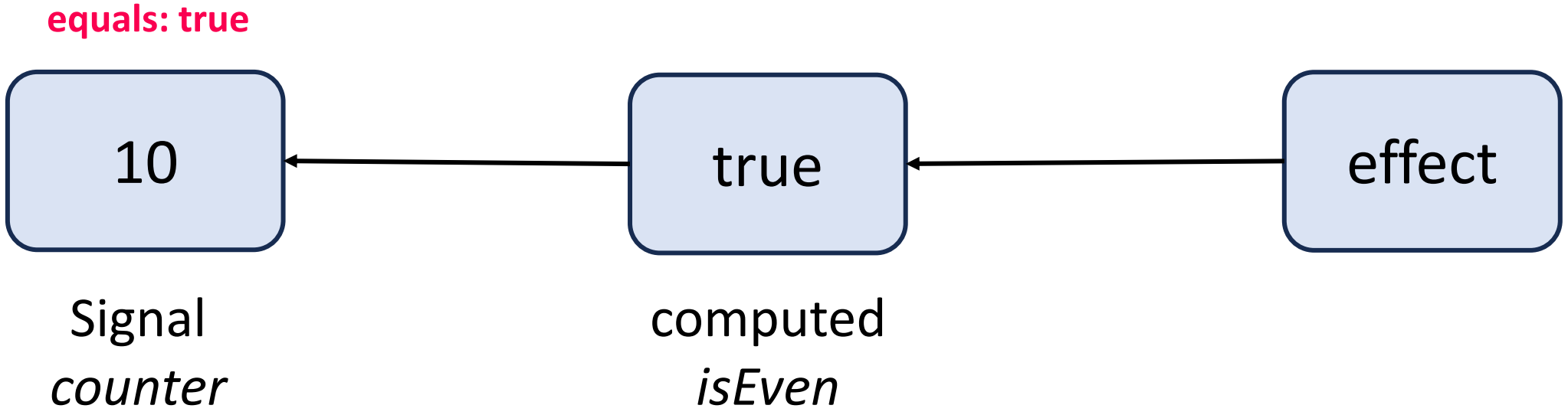
```
@Injectable({ providedIn: 'root' })
export class FlightBookingFacade {
  […]
  private state = signal({
    from: 'Hamburg',
    to: 'Graz',
    flights: [] as Flight[], […]
  }, { equal });

  flights = computed(() => this.state().flights, { equal });
  from = computed(() => this.state().from, { equal });
  […]
}
```

Reactive Chain with Signals



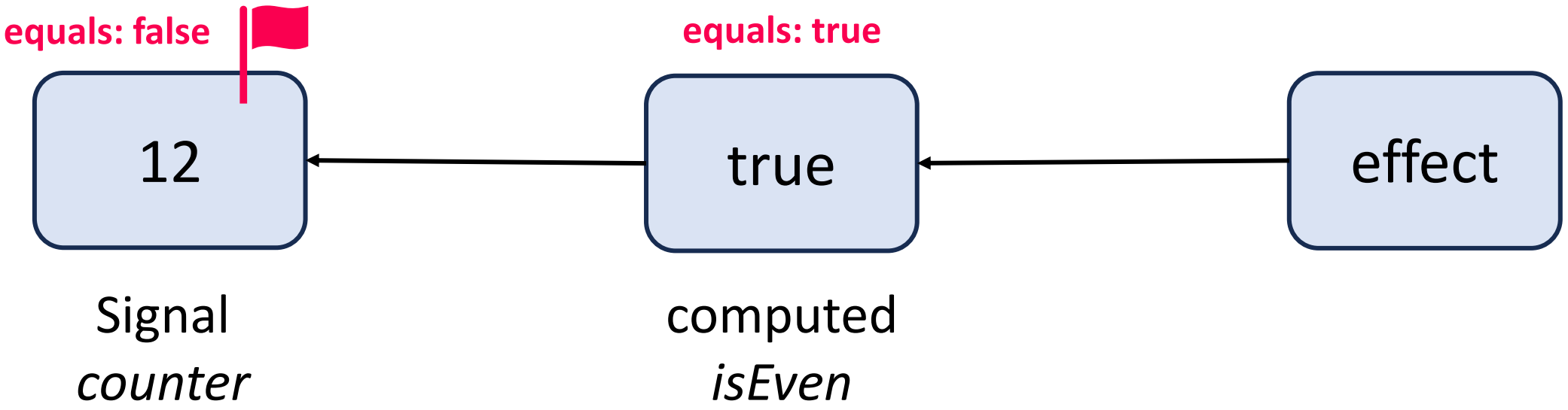
Reactive Chain with Signals

counter.set(10)

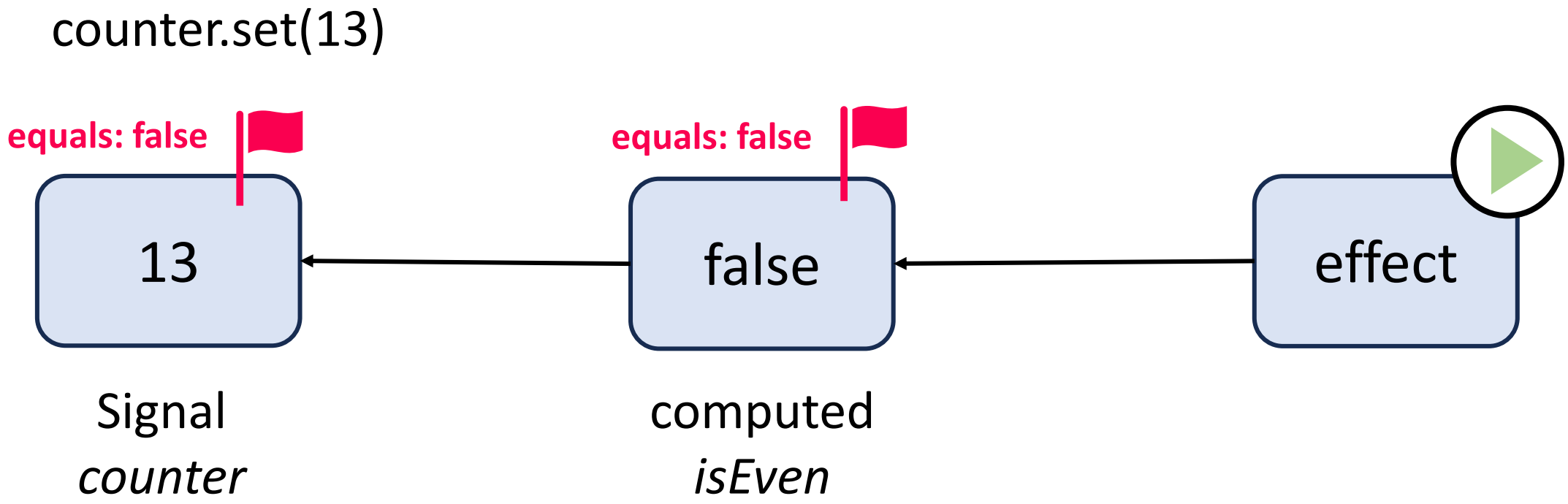


Reactive Chain with Signals

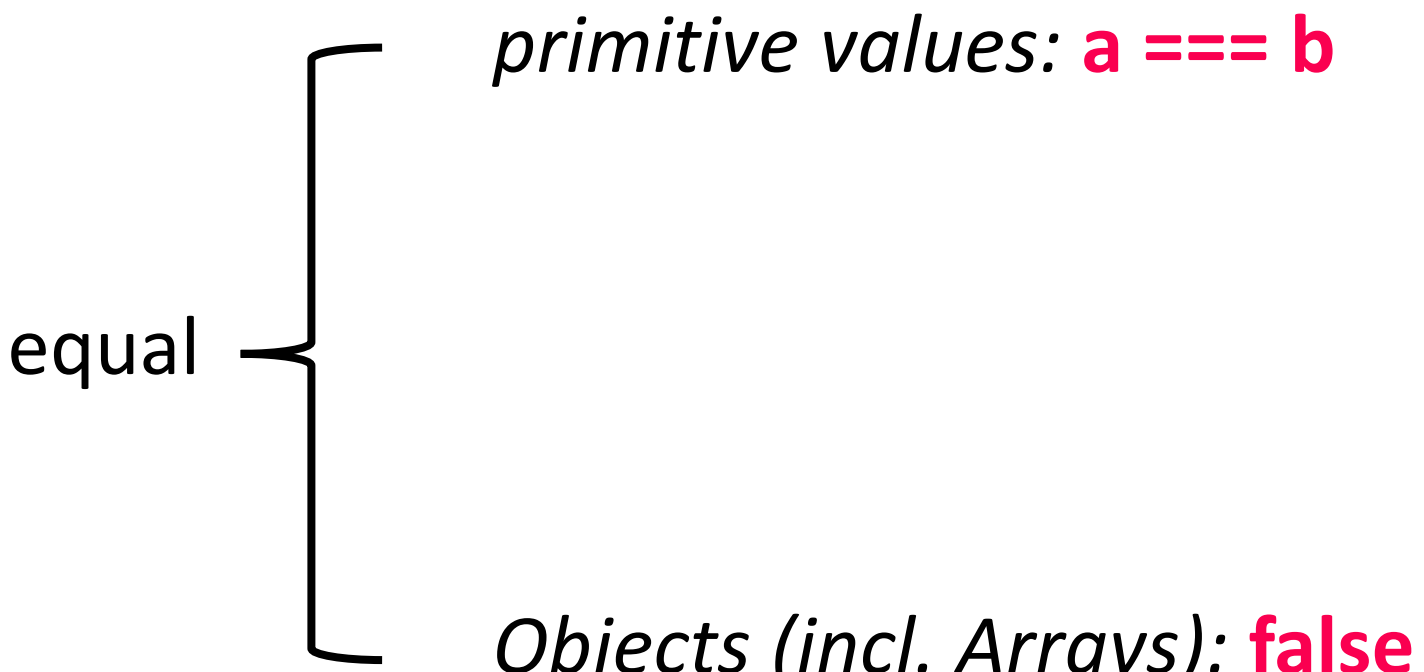
counter.set(12)



Reactive Chain with Signals



equal: Default Behavior



equal for Immutable Data

equal → **a === b**

State Signal

```
const equal = (a,b) => a === b
```

```
@Injectable({ providedIn: 'root' })
export class FlightBookingFacade {
  […]
  private state = signal({
    from: 'Hamburg',
    to: 'Graz',
    flights: [] as Flight[], […]
  }, { equal });

  flights = computed(() => this.state().flights, { equal });
  from = computed(() => this.state().from, { equal });
  […]
}
```

DEMO

(branch: arc-facade-3a)

ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

DEMO

(branch: arc-facade-3d)

ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

#3:
NGRX + Signals

ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

Selecting Signals

select(selector)
selectSignal(selector)

DEMO

(branch: arc-ngrx)

ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

#4:
The Upcoming NGRX Signal Store

ANGULAR ARCHITECTS
INSIDE KNOWLEDGE

Signal Store (Flavor 1)

```
@Injectable({ providedIn: 'root' })
export class FlightBookingFacade {

  private state = signalState({
    from: 'Paris',
    to: 'London',
    flights: [] as Flight[],
    basket: {} as Record<number, boolean>,
  });

  flights = this.state.flights;
  from = this.state.from;

  […]
}
```

Signal Store (Flavor 1)

```
@Injectable({ providedIn: 'root' })
export class FlightBookingFacade {
  […]

  selected = selectSignal(
      () => this.flights().filter((f) => this.basket()[f.id]);

  […]
}
```

Signal Store (Flavor 1)

```
@Injectable({ providedIn: 'root' })
export class FlightBookingFacade {
  […]

  selected2 = selectSignal(
    this.flights,
    this.basket,
    (flights, basket) => flights.filter((f) => basket[f.id])
  );

  […]
}
```

Signal Store (Flavor 1)

```
@Injectable({ providedIn: 'root' })
export class FlightBookingFacade {
  […]

  updateCriteria(from: string, to: string): void {

    this.state.$update({ from, to });

  }

  […]
}
```

Signal Store (Flavor 1)

```
@Injectable({ providedIn: 'root' })
export class FlightBookingFacade {
  […]

  updateCriteria(from: string, to: string): void {

    this.state.$update(state => ({ ...state, from, to }));

  }

  […]
}
```

Signal Store (Flavor 1)

```
@Injectable({ providedIn: 'root' })
export class FlightBookingFacade {
  […]

  updateCriteria(from: string, to: string): void {

    this.state.$update(updateRoute(from, to));

  }

  […]
}
```

```
function updateRoute<T>(from: string, to: string) {

  return (state: T) => ({ ...state, from, to })

}
```

DEMO

(branch: arc-signal-store)

ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

Signal Store: Flavor 2

```
export const FlightBookingStore = signalStore(
  { providedIn: 'root' },
  […]
);
```

Signal Store: Flavor 2

```
export const FlightBookingStore = signalStore(
  { providedIn: 'root' },
  withState({
    from: 'Paris',
    to: 'London',
    […]
  }),
  […]
);
```

Signal Store: Flavor 2

```
export const FlightBookingStore = signalStore(
  { providedIn: 'root' },
  withState({
    from: 'Paris',
    to: 'London',
    […]
  }),
  withSignals(([…]) => ({ […] })),
  withMethods(([…]) => ({ })),
  withHooks({ […] }),
  withCallState()
);
```

DEMO

(branch: arc-signal-store-2)

ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

Signal Store: Flavor 3

```
const BooksStore = signalStore(
  withEntities<Book>({ collection: 'book' }),
  withEntities<Author>({ collection: 'author' })
);
```

Signal Store: Flavor 3

```
const BooksStore = signalStore(
  withLoadEntities<Book>(BookService),
  withLoadEntities<Author>(AuthorService),
);
```

Conclusion

Services + Signals	equal	NGRX
NGRX Signal Store	Different Flavors	Custom Features