

## When Should We (NOT) Use a Signal effect()?



Deborah Kurata  
19.6K subscribers

Subscribe

408



Share

Download

Thanks



5,735 views Dec 4, 2024 #angular #effect #resource

"An effect is an operation that runs whenever one or more signal values change." Does that mean we should use it anytime we want to react to changes in a signal?

Angular v19 includes changes to effect, and introduces several additional signal features. Does that change the guidance?

In this video, we walk through several scenarios, examining an effect() based approach and an alternate approach. Then we look at the version 19 changes to the effect() function and summarize suggested guidelines.

### Content

00:00 effect()

00:39 effect() vs computed()

02:38 effect() vs linkedSignal()

04:34 effect() vs rxResource()

06:34 effect() and logging

07:06 Changes to effect() in v19

08:34 Recommendation for using effect()

# effect()

"An operation that runs whenever one or more signal values change."  
- Angular docs

### Star Wars Vehicle Sales

--Select a vehicle--  
Quantity:   
Vehicle:  
Movie:  
Price:  
Total: 0

### Star Wars Vehicle Sales

AT-AT  
Quantity:   
Vehicle: AT-AT  
Movie: The Empire Strikes Back  
Price: 10050  
Total: 10050

```
1 import { Component, computed, effect, inject, linkedSignal, signal } from '@angular/core';
2 import { FormsModule } from '@angular/forms';
3 import { bootstrapApplication } from '@angular/platform-browser';
4 import { rxResource } from '@angular/core/rxjs-interop';
5 import { HttpClient, provideHttpClient } from '@angular/common/http';
6
7
8 @Component({
9   selector: 'app-root',
10  imports: [FormsModule],
11  template: `
12    <h1>Star Wars Vehicle Sales</h1>
13    <select class="select" (change)="onSelectedVehicle($event.target)">
14      <option value="" disabled selected>--Select a vehicle--</option>
15      @for(vehicle of vehicles(); track vehicle) {
16        <option [value]="vehicle.id">{{ vehicle.name }}</option>
17      }
18    </select>
19    <div>Quantity: <input type="number" [(ngModel)]="quantity" ></div>
20    <div>Vehicle: {{ selectedVehicle()?.name }}</div>
21    <div>Movie: {{ movie()?.title }}</div>
22    <div>Price: {{ selectedVehicle()?.price }}</div>
23    <div [style.color]="color()">Total: {{ total() }}</div>
24  `
25 })
26 export class App {
27   private url = 'https://swapi.py4e.com/api/films';
28
29   // Signals to support the template
30   selectedVehicle = signal<Vehicle | undefined>(undefined);
```

StackBlitz editor showing a TypeScript file named `main.ts` and a preview of a web application titled "Star Wars Vehicle Sales".

**main.ts Code:**

```
38 // Task 1: React to changes and adjust the total and color.
39 total = signal(0);
40 color = signal('blue');
41 totEff = effect(() => {
42   this.total.set((this.selectedVehicle()?.price ?? 0) * this.quantity());
43   this.color.set(this.total() > 50000 ? 'green' : 'blue');
44 });
45
46 // Task 2: Reset the quantity when the vehicle changes
47 // qtyResetEff = effect(() => {
48 //   if (this.selectedVehicle()) {
49 //     this.quantity.set(1);
50 //   }
51 // });
52
53 // Task 3: Retrieve the movies for the selected vehicle
54 http = inject(HttpClient);
55 movie = signal<Film | undefined>(undefined);
56 movieEff = effect(() => this.http.get<Film>(`${this.url}/${this.selectedVehicle()?.id}`)
57   .subscribe(m => this.movie.set(m)));
58
59 // Task 4: Log out signals when they change
60 qtyEff = effect(() => console.log('quantity:', this.quantity()));
61 vehEff = effect(() => console.log('vehicle:', JSON.stringify(this.selectedVehicle())));
62
63 onSelectedVehicle(ele: EventTarget | null) {
64   // Get the id from the element
65   const id = Number((ele as HTMLSelectElement).value);
66   // Find the vehicle in the array
```

**Star Wars Vehicle Sales Preview:**

--Select a vehicle--  
Quantity: 1  
Vehicle:  
Movie:  
Price:  
Total: 0

StackBlitz editor showing the same TypeScript file and preview. The dropdown menu for "Select a vehicle" is open, showing options: Sand Crawler, AT-AT, and TIE Fighter. A yellow arrow points to the "Sand Crawler" option.

**main.ts Code:**

```
38 // Task 1: React to changes and adjust the total and color.
39 total = signal(0);
40 color = signal('blue');
41 totEff = effect(() => {
42   this.total.set((this.selectedVehicle()?.price ?? 0) * this.quantity());
43   this.color.set(this.total() > 50000 ? 'green' : 'blue');
44 });
45
46 // Task 2: Reset the quantity when the vehicle changes
47 // qtyResetEff = effect(() => {
48 //   if (this.selectedVehicle()) {
```

**Star Wars Vehicle Sales Preview:**

--Select a vehicle--  
--Select a vehicle--  
Sand Crawler  
AT-AT  
TIE Fighter

StackBlitz editor showing the same TypeScript file and preview. The dropdown menu for "Select a vehicle" is closed, and "AT-AT" is selected. The quantity is 4. A yellow arrow points to the "AT-AT" option in the dropdown menu.

**main.ts Code:**

```
38 // Task 1: React to changes and adjust the total and color.
39 total = signal(0);
40 color = signal('blue');
41 totEff = effect(() => {
42   this.total.set((this.selectedVehicle()?.price ?? 0) * this.quantity());
43   this.color.set(this.total() > 50000 ? 'green' : 'blue');
44 });
45
46 // Task 2: Reset the quantity when the vehicle changes
47 // qtyResetEff = effect(() => {
48 //   if (this.selectedVehicle()) {
```

**Star Wars Vehicle Sales Preview:**

AT-AT  
Quantity: 4  
Vehicle: AT-AT  
Movie: The Empire Strikes Back  
Price: 10050  
Total: 40200

StackBlitz editor showing the same TypeScript file and preview. The dropdown menu for "Select a vehicle" is closed, and "AT-AT" is selected. The quantity is 5. A yellow arrow points to the "AT-AT" option in the dropdown menu.

**main.ts Code:**

```
38 // Task 1: React to changes and adjust the total and color.
39 total = signal(0);
40 color = signal('blue');
41 totEff = effect(() => {
42   this.total.set((this.selectedVehicle()?.price ?? 0) * this.quantity());
43   this.color.set(this.total() > 50000 ? 'green' : 'blue');
44 });
45
46 // Task 2: Reset the quantity when the vehicle changes
47 // qtyResetEff = effect(() => {
48 //   if (this.selectedVehicle()) {
49 //     this.quantity.set(1);
```

**Star Wars Vehicle Sales Preview:**

AT-AT  
Quantity: 5  
Vehicle: AT-AT  
Movie: The Empire Strikes Back  
Price: 10050  
Total: 50250

```
38 // Task 1: React to changes and adjust the total and color.
39 total = signal(0);
40 color = signal('blue');
41 totEff = effect(() => {
42   this.total.set((this.selectedVehicle()?.price ?? 0) * this.quantity());
43   this.color.set(this.total() > 50000 ? 'green' : 'blue');
44 });
45
46 // Task 2: Reset the quantity when the vehicle changes
47 // qtyResetEff = effect(() => {
48 //   if (this.selectedVehicle()) {
49 //     this.quantity.set(1);
50 //   }
51 // });
```

### Star Wars Vehicle Sales

AT-AT  
Quantity: 5  
Vehicle: AT-AT  
Movie: The Empire Strikes Back  
Price: 10050  
Total: 50250

```
46 // Task 2: Reset the quantity when the vehicle changes
47 // qtyResetEff = effect(() => {
48 //   if (this.selectedVehicle()) {
49 //     this.quantity.set(1);
50 //   }
51 // });
52
53 // Task 3: Retrieve the movies for the selected vehicle
54 http = inject(HttpClient);
55 movie = signal<Film | undefined>(undefined);
56 movieEff = effect(() => this.http.get<Film>(`${this.url}/${this.selectedVehicle()?.id}`)
57   .subscribe(m => this.movie.set(m)));
58
59 // Task 4: Log out signals when they change
60 qtyEff = effect(() => console.log('quantity:', this.quantity()));
61 vehEff = effect(() => console.log('vehicle:', JSON.stringify(this.selectedVehicle())));
62
63 onSelectedVehicle(ele: EventTarget | null) {
64   // Get the id from the element
65   const id = Number((ele as HTMLSelectElement).value);
66   // Find the vehicle in the array
```

### Star Wars Vehicle Sales

AT-AT  
Quantity: 5  
Vehicle: AT-AT  
Movie: The Empire Strikes Back  
Price: 10050  
Total: 50250

```
46 // Task 2: Reset the quantity when the vehicle changes
47 // qtyResetEff = effect(() => {
48 //   if (this.selectedVehicle()) {
49 //     this.quantity.set(1);
50 //   }
51 // });
52
53 // Task 3: Retrieve the movies for the selected vehicle
54 http = inject(HttpClient);
55 movie = signal<Film | undefined>(undefined);
56 movieEff = effect(() => this.http.get<Film>(`${this.url}/${this.selectedVehicle()?.id}`)
57   .subscribe(m => this.movie.set(m)));
58
59 // Task 4: Log out signals when they change
60 qtyEff = effect(() => console.log('quantity:', this.quantity()));
61 vehEff = effect(() => console.log('vehicle:', JSON.stringify(this.selectedVehicle())));
62
63 onSelectedVehicle(ele: EventTarget | null) {
64   // Get the id from the element
65   const id = Number((ele as HTMLSelectElement).value);
66   // Find the vehicle in the array
```

### Star Wars Vehicle Sales

AT-AT  
Quantity: 5  
Vehicle: AT-AT  
Movie: The Empire Strikes Back  
Price: 10050  
Total: 50250

There is a much better way to accomplish this task using **computed()** signals as below



stackblitz.com/~edit/effect-guidance-deborahk-n7sogg?file=src/main.ts

effect() Guidance (start) (forked)

Share

TS main.ts

41

42

43

44

45

46

47

48

49

50

51

52

53

54

55

56

57

58

**computed()**  
Dependency tracking  
Lazy evaluation  
Memoization  
Declarative coding

// Task 1: React to changes and adjust the total and color.  
total = computed(() => (this.selectedVehicle()?.price ?? 0) \* this.quantity());  
color = computed(() => this.total() > 50000 ? 'green' : 'blue');

// Task 2: Reset the quantity when the vehicle changes  
// qtyResetEff = effect(() => {  
// if (this.selectedVehicle()) {  
// this.quantity.set(1);  
// }  
// });

// Task 3: Retrieve the movies for the selected vehicle  
http = inject(HttpClient);  
movie = signal<Film | undefined>(undefined);  
movieEff = effect(() => this.http.get<Film>(`\${this.url}/\${this.selectedVehicle()?.id}`)  
 .subscribe(m => this.movie.set(m)));

// Task 4: Log out signals when they change  
qtyEff = effect(() => console.log('quantity:', this.quantity()));

Star Wars Vehicle Sales

--Select a vehicle--  
Quantity: 1  
Vehicle:  
Movie:  
Price:  
Total: 0

stackblitz.com/~edit/effect-guidance-deborahk-n7sogg?file=src/main.ts

effect() Guidance (start) (forked)

Share

TS main.ts

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

45

46

47

48

49

50

51

52

53

54

55

56

// Signals to support the template  
selectedVehicle = signal<Vehicle | undefined>(undefined);  
quantity = signal<number>(1);

vehicles = signal<Vehicle[]>([  
 { id: 1, name: 'Sand Crawler', price: 22050 },  
 { id: 2, name: 'AT-AT', price: 10050 },  
 { id: 3, name: 'TIE Fighter', price: 55000 }  
]);

// Task 1: React to changes and adjust the total and color.  
total = computed(() => (this.selectedVehicle()?.price ?? 0) \* this.quantity());  
color = computed(() => this.total() > 50000 ? 'green' : 'blue');

// Task 2: Reset the quantity when the vehicle changes  
// qtyResetEff = effect(() => {  
// if (this.selectedVehicle()) {  
// this.quantity.set(1);  
// }  
// });

// Task 3: Retrieve the movies for the selected vehicle  
http = inject(HttpClient);  
movie = signal<Film | undefined>(undefined);  
movieEff = effect(() => this.http.get<Film>(`\${this.url}/\${this.selectedVehicle()?.id}`)  
 .subscribe(m => this.movie.set(m)));

// Task 4: Log out signals when they change  
qtyEff = effect(() => console.log('quantity:', this.quantity()));

Star Wars Vehicle Sales

--Select a vehicle--  
Quantity: 1  
Vehicle:  
Movie:  
Price:  
Total: 0

stackblitz.com/~edit/effect-guidance-deborahk-n7sogg?file=src/main.ts

effect() Guidance (start) (forked)

Share

TS main.ts

28

29

30

31

32

33

34

35

36

37

38

39

40

41

(property) App.selectedVehicle: WritableSignal<Vehicle | undefined>

selectedVehicle = signal<Vehicle | undefined>(undefined);  
quantity = signal<number>(1);

vehicles = signal<Vehicle[]>([  
 { id: 1, name: 'Sand Crawler', price: 22050 },  
 { id: 2, name: 'AT-AT', price: 10050 },  
 { id: 3, name: 'TIE Fighter', price: 55000 }  
]);

// Task 1: React to changes and adjust the total and color.  
total = computed(() => (this.selectedVehicle()?.price ?? 0) \* this.quantity());  
color = computed(() => this.total() > 50000 ? 'green' : 'blue');

Star Wars Vehicle Sales

--Select a vehicle--  
Quantity: 1  
Vehicle:  
Movie:  
Price:  
Total: 0

```
28 // Signals to support the template
29 selectedVehicle = signal<Vehicle | undefined>(undefined);
30 quantity = signal<number>(1);
31
32 vehicles = signal<Vehicle[]>([
33   { id: 1, name: 'Sand Crawler', price: 22050 },
34   { id: 2, name: 'AT-AT', price: 10050 },
35   { id: 3, name: 'TIE Fighter', price: 55000 }
36 ]);
37
38 // Task 1: React to changes and adjust the total and color.
39 total = computed(() => (this.selectedVehicle()?.price ?? 0) * this.quantity());
40 color = computed(() => this.total() > 50000 ? 'green' : 'blue');
41
42 // Task 2: Reset the quantity when the vehicle changes
43 // qtyResetEff = effect(() => {
44 //   if (this.selectedVehicle()) {
45 //     this.quantity.set(1);
46 //   }
47 // });
48
49 // Task 3: Retrieve the movies for the selected vehicle
50 http = inject(HttpClient);
51 movie = signal<Film | undefined>(undefined);
52 movieEff = effect(() => this.http.get<Film>(`${this.url}/${this.selectedVehicle()?.id}`)
53   .subscribe(m => this.movie.set(m)));
54
55 // Task 4: Log out signals when they change
56 qtyEff = effect(() => console.log('quantity:', this.quantity()));
```

### Star Wars Vehicle Sales

--Select a vehicle--  
Quantity: 1  
Vehicle: Sand Crawler  
Movie:  
Price: 22050  
Total: 0

Next, let us see how to update

```
28 // Signals to support the template
29 selectedVehicle = signal<Vehicle | undefined>(undefined);
30 quantity = signal<number>(1);
31
32 vehicles = signal<Vehicle[]>([
33   { id: 1, name: 'Sand Crawler', price: 22050 },
34   { id: 2, name: 'AT-AT', price: 10050 },
35   { id: 3, name: 'TIE Fighter', price: 55000 }
36 ]);
37
38 // Task 1: React to changes and adjust the total and color.
39 total = computed(() => (this.selectedVehicle()?.price ?? 0) * this.quantity());
40 color = computed(() => this.total() > 50000 ? 'green' : 'blue');
41
42 // Task 2: Reset the quantity when the vehicle changes
43 // qtyResetEff = effect(() => {
44 //   if (this.selectedVehicle()) {
45 //     this.quantity.set(1);
46 //   }
47 // });
48
49 // Task 3: Retrieve the movies for the selected vehicle
50 http = inject(HttpClient);
51 movie = signal<Film | undefined>(undefined);
52 movieEff = effect(() => this.http.get<Film>(`${this.url}/${this.selectedVehicle()?.id}`)
53   .subscribe(m => this.movie.set(m)));
54
55 // Task 4: Log out signals when they change
56 qtyEff = effect(() => console.log('quantity:', this.quantity()));
```

### Star Wars Vehicle Sales

AT-AT  
Quantity: 5  
Vehicle: AT-AT  
Movie: The Empire Strikes Back  
Price: 10050  
Total: 50250

```
28 // Signals to support the template
29 selectedVehicle = signal<Vehicle | undefined>(undefined);
30 quantity = signal<number>(1);
31
32 vehicles = signal<Vehicle[]>([
33   { id: 1, name: 'Sand Crawler', price: 22050 },
34   { id: 2, name: 'AT-AT', price: 10050 },
35   { id: 3, name: 'TIE Fighter', price: 55000 }
36 ]);
37
```

### Star Wars Vehicle Sales

AT-AT  
--Select a vehicle--  
Sand Crawler  
AT-AT  
TIE Fighter  
Strikes Back

stackblitz.com/~edit/effect-guidance-deborahk-n7sogg?file=src/main.ts

effect() Guidance (start) (forked)

Share

main.ts

```
28 // Signals to support the template
29 selectedVehicle = signal<Vehicle | undefined>(undefined);
30 quantity = signal<number>(1);
31
32 vehicles = signal<Vehicle[]>([
33   { id: 1, name: 'Sand Crawler', price: 22050 },
34   { id: 2, name: 'AT-AT', price: 10050 },
35   { id: 3, name: 'TIE Fighter', price: 55000 }
36 ]);
37
38 // Task 1: React to changes and adjust the total and color.
39 total = computed(() => (this.selectedVehicle()?.price ?? 0) * this.quantity());
40 color = computed(() => this.total() > 50000 ? 'green' : 'blue');
41
42 // Task 2: Reset the quantity when the vehicle changes
43 qtyResetEff = effect(() => {
44   if (this.selectedVehicle()) {
45     this.quantity.set(1);
46   }
47 });
48
49 // Task 3: Retrieve the movies for the selected vehicle
50 http = inject(HttpClient);
51 movie = signal<Film | undefined>(undefined);
52 movieEff = effect(() => this.http.get<Film>(`${this.url}/${this.selectedVehicle()?.id}`)
53   .subscribe(m => this.movie.set(m)));
54
55 // Task 4: Log out signals when they change
56 qtyEff = effect(() => console.log('quantity:', this.quantity()));
```

## Star Wars Vehicle Sales

TIE Fighter

Quantity: 5

Vehicle: TIE Fighter

Movie: Return of the Jedi

Price: 55000

Total: 275000

stackblitz.com/~edit/effect-guidance-deborahk-n7sogg?file=src/main.ts

effect() Guidance (start) (forked)

Share

main.ts

```
28 // Signals to support the template
29 selectedVehicle = signal<Vehicle | undefined>(undefined);
30 quantity = signal<number>(1);
31
32 vehicles = signal<Vehicle[]>([
33   { id: 1, name: 'Sand Crawler', price: 22050 },
34   { id: 2, name: 'AT-AT', price: 10050 },
35   { id: 3, name: 'TIE Fighter', price: 55000 }
36 ]);
37
38 // Task 1: React to changes and adjust the total and color.
39 total = computed(() => (this.selectedVehicle()?.price ?? 0) * this.quantity());
40 color = computed(() => this.total() > 50000 ? 'green' : 'blue');
41
42 // Task 2: Reset the quantity when the vehicle changes
43 qtyResetEff = effect(() => {
44   if (this.selectedVehicle()) {
45     this.quantity.set(1);
46   }
47 });
48
49 // Task 3: Retrieve the movies for the selected vehicle
50 http = inject(HttpClient);
51 movie = signal<Film | undefined>(undefined);
52 movieEff = effect(() => this.http.get<Film>(`${this.url}/${this.selectedVehicle()?.id}`)
53   .subscribe(m => this.movie.set(m)));
54
55 // Task 4: Log out signals when they change
56 qtyEff = effect(() => console.log('quantity:', this.quantity()));
```

## Star Wars Vehicle Sales

AT-AT

Quantity: 5

Vehicle: AT-AT

Movie: The Empire Strikes Back

Price: 10050

Total: 50250



```
28 // Signals to support the template
29 selectedVehicle = signal<Vehicle | undefined>(undefined);
30 quantity = signal<number>(1);
31
32 vehicles = signal<Vehicle[]>([
33   { id: 1, name: 'Sand Crawler', price: 22050 },
34   { id: 2, name: 'AT-AT', price: 10050 },
35   { id: 3, name: 'TIE Fighter', price: 55000 }
36 ]);
37
38 // Task 1: React to changes and adjust the total and color.
39 total = computed(() => (this.selectedVehicle()?.price ?? 0) * this.quantity());
40 color = computed(() => this.total() > 50000 ? 'green' : 'blue');
41
42 // Task 2: Reset the quantity when the vehicle changes
43 qtyResetEff = effect(() => {
44   if (this.selectedVehicle()) {
45     this.quantity.set(1);
46   }
47 });
48
49 // Task 3: Retrieve the movies for the selected vehicle
50 http = inject(HttpClient);
51 movie = signal<Film | undefined>(undefined);
52 movieEff = effect(() => this.http.get<Film>(`${this.url}/${this.selectedVehicle()?.id}`)
53   .subscribe(m => this.movie.set(m)));
54
55 // Task 4: Log out signals when they change
56 qtyEff = effect(() => console.log('quantity:', this.quantity()));
```

**Star Wars Vehicle Sales**

TIE Fighter  
Quantity: 1  
Vehicle: TIE Fighter  
Movie: Return of the Jedi  
Price: 55000  
Total: 55000

It works. But there is a better way starting in version 19 where we now have **LinkedSignal**

```
28 // Signals to support the template
29 selectedVehicle = signal<Vehicle | undefined>(undefined);
30 quantity = linkedSignal({
31   source: this.selectedVehicle,
32   computation: () => 1
33 });
34
35 vehicles = signal<Vehicle[]>([
36   { id: 1, name: 'Sand Crawler', price: 22050 },
37   { id: 2, name: 'AT-AT', price: 10050 },
38   { id: 3, name: 'TIE Fighter', price: 55000 }
39 ]);
40
41 // Task 1: React to changes and adjust the total and color.
42 total = computed(() => (this.selectedVehicle()?.price ?? 0) * this.quantity());
43 color = computed(() => this.total() > 50000 ? 'green' : 'blue');
44
45 // Task 2: Reset the quantity when the vehicle changes
46
47 // Task 3: Retrieve the movies for the selected vehicle
48 http = inject(HttpClient);
49 movie = signal<Film | undefined>(undefined);
50 movieEff = effect(() => this.http.get<Film>(`${this.url}/${this.selectedVehicle()?.id}`)
51   .subscribe(m => this.movie.set(m)));
52
53 // Task 4: Log out signals when they change
54 qtyEff = effect(() => console.log('quantity:', this.quantity()));
55 vehEff = effect(() => console.log('vehicle:', JSON.stringify(this.selectedVehicle())));
56
```

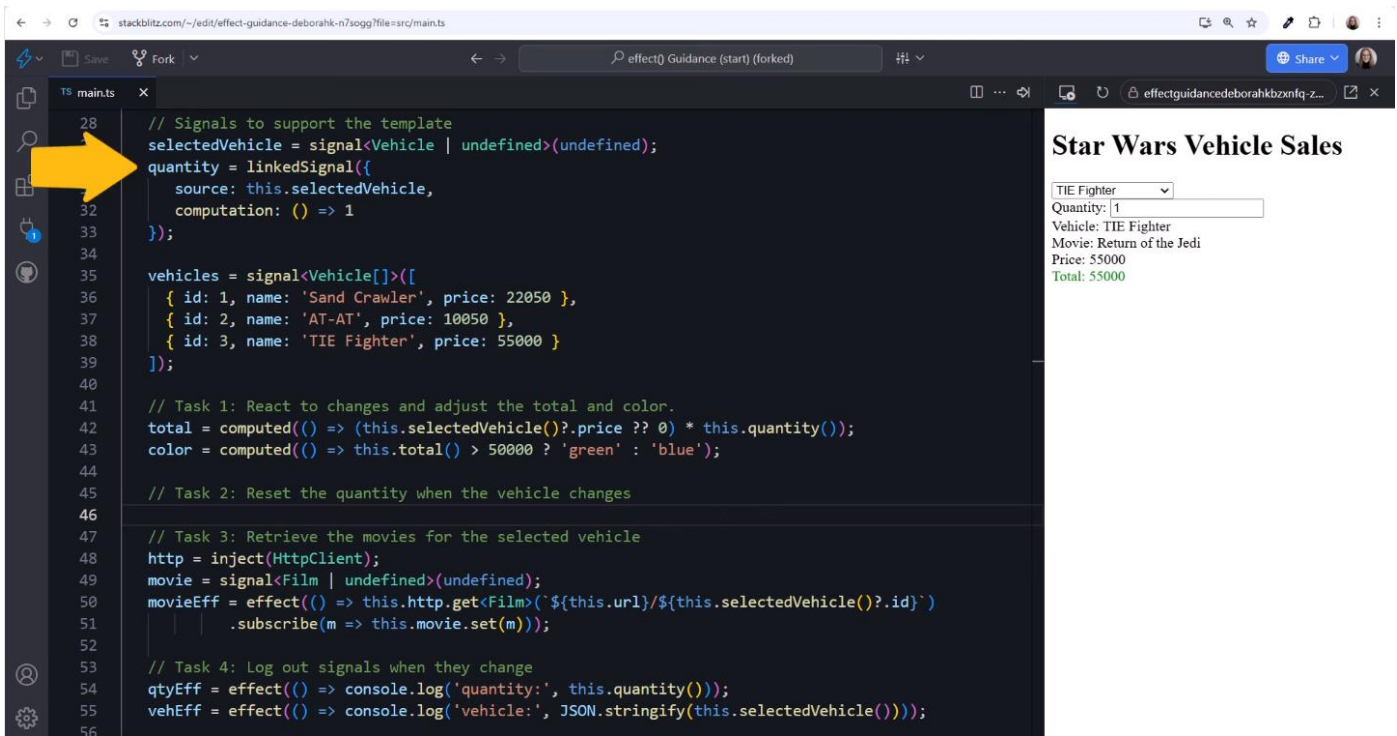
**Star Wars Vehicle Sales**

TIE Fighter  
Quantity: 1  
Vehicle: TIE Fighter  
Movie: Return of the Jedi  
Price: 55000  
Total: 55000

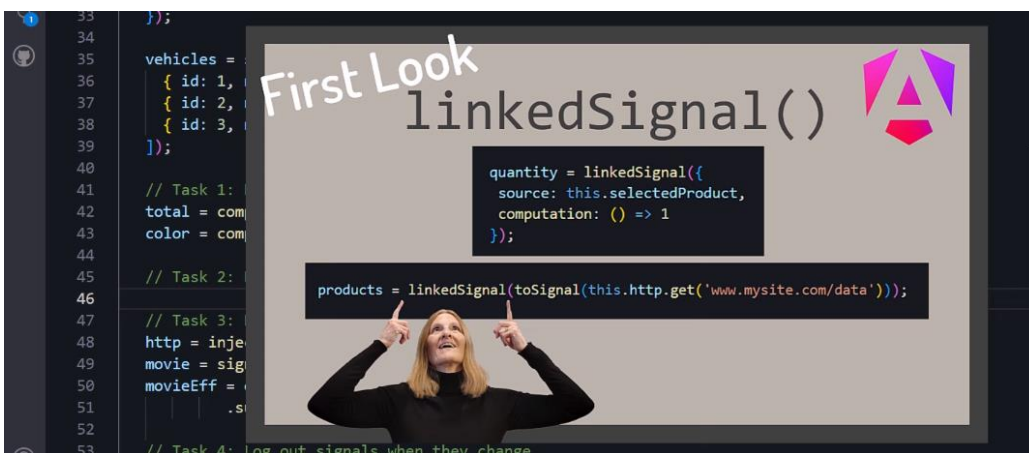
```
28 // Signals to support the template
29 selectedVehicle = signal<Vehicle | undefined>(undefined);
30 quantity = linkedSignal({
31   source: this.selectedVehicle,
32   computation: () => 1
33 });
34
35 vehicles = signal<Vehicle[]>([
36   { id: 1, name: 'Sand Crawler', price: 22050 },
37   { id: 2, name: 'AT-AT', price: 10050 },
38   { id: 3, name: 'TIE Fighter', price: 55000 }
39 ]);
40
```

**Star Wars Vehicle Sales**

TIE Fighter  
Quantity: 1  
Vehicle: TIE Fighter  
Movie: Return of the Jedi  
Price: 55000  
Total: 55000



No need for an effect here anymore!





Star Wars Vehicle Sales

AT-AT

Quantity: 1

Vehicle: AT-AT

Movie: The Empire Strikes Back

Price: 10050

Total: 10050

```
// Signals to support the template
selectedVehicle = signal<Vehicle | undefined>(undefined);
quantity = linkedSignal({
  source: this.selectedVehicle,
  computation: () => 1
});

vehicles = signal<Vehicle[]>([
  { id: 1, name: 'Sand Crawler', price: 22050 },
  { id: 2, name: 'AT-AT', price: 10050 },
  { id: 3, name: 'TIE Fighter', price: 55000 }
]);

// Task 1: React to changes and adjust the total and color.
total = computed(() => (this.selectedVehicle()?.price ?? 0) * this.quantity());
color = computed(() => this.total() > 50000 ? 'green' : 'blue');

// Task 2: Reset the quantity when the vehicle changes

// Task 3: Retrieve the movies for the selected vehicle
http = inject(HttpClient);
movie = signal<Film | undefined>(undefined);
movieEff = effect(() => this.http.get<Film>(`${this.url}/${this.selectedVehicle()?.id}`)
  .subscribe(m => this.movie.set(m)));

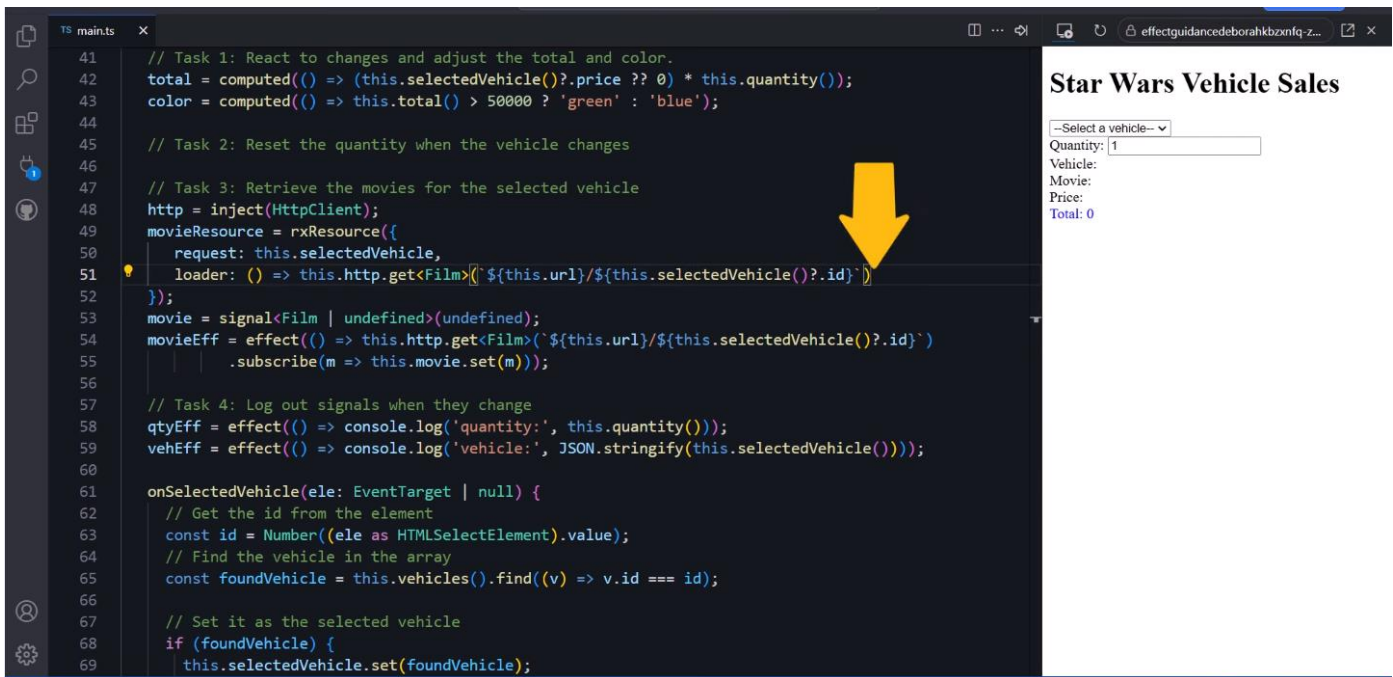
// Task 4: Log out signals when they change
qtyEff = effect(() => console.log('quantity:', this.quantity()));
vehEff = effect(() => console.log('vehicle:', JSON.stringify(this.selectedVehicle())));
```

We can use the new **rxResource()** here instead of using an effect and having to manage the subscriptions.

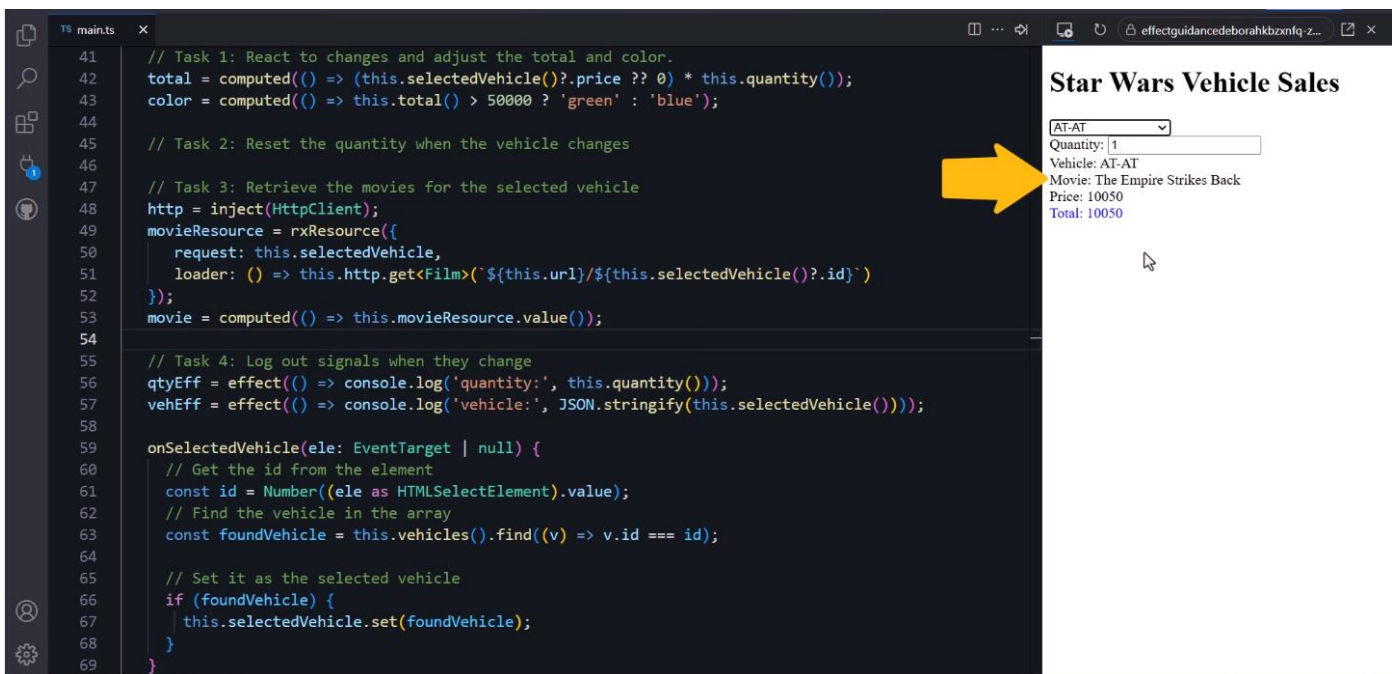
```
import { Component, computed, effect, inject, linkedSignal, signal } from '@angular/core';
import { FormsModule } from '@angular/forms';
import { bootstrapApplication } from '@angular/platform-browser';
import { rxResource } from '@angular/core/rxjs-interop';
import { HttpClient, provideHttpClient } from '@angular/common/http';

@Component({
  selector: 'app-root',
  imports: [FormsModule],
  template: `
    <h1>Star Wars Vehicle Sales</h1>
    <select class="select" (change)="onSelectedVehicle($event.target)">
      <option value="" disabled selected>--Select a vehicle--</option>
      @for(vehicle of vehicles(); track vehicle) {
        <option [value]="vehicle.id">{{ vehicle.name }}</option>
      }
    </select>
    <div>Quantity: <input type="number" [(ngModel)]="quantity" ></div>
    <div>Vehicle: {{ selectedVehicle()?.name }}</div>
    <div>Movie: {{ movie()?.title }}</div>
    <div>Price: {{ selectedVehicle()?.price }}</div>
    <div [style.color]="color()">Total: {{ total() }}</div>
  `
})
export class App {
  private url = 'https://swapi.py4e.com/api/films';

  // Signals to support the template
  selectedVehicle = signal<Vehicle | undefined>(undefined);
```



Notice that we don't have to subscribe anymore since `rxResource()` is managing it for us



Next, let us see the effect use-case for logging out signals as they change

```
stackblitz.com/~edit/effect-guidance-deborahk-n7sogg?file=src/main.ts
effect() Guidance (start) (forked)
Share
TS main.ts
41 // Task 1: React to changes and adjust the total and color.
42 total = computed(() => (this.selectedVehicle()?.price ?? 0) * this.quantity());
43 color = computed(() => this.total() > 50000 ? 'green' : 'blue');
44
45 // Task 2: Reset the quantity when the vehicle changes
46
47 // Task 3: Retrieve the movies for the selected vehicle
48 http = inject(HttpClient);
49 movieResource = rxResource({
50   request: this.selectedVehicle,
51   loader: () => this.http.get<Film>(`${this.url}/${this.selectedVehicle()?.id}`)
52 });
53 movie = computed(() => this.movieResource.value());
54
55 // Task 4: Log out signals when they change
56 qtyEff = effect(() => console.log('quantity:', this.quantity()));
57 vehEff = effect(() => console.log('vehicle:', JSON.stringify(this.selectedVehicle())));
58
59 onSelectVehicle(ele: EventTarget | null) {
60   // Get the id from the element
61   const id = Number((ele as HTMLSelectElement).value);
62   // Find the vehicle in the array
63   const foundVehicle = this.vehicles().find((v) => v.id === id);
64
65   // Set it as the selected vehicle
66   if (foundVehicle) {
67     this.selectedVehicle.set(foundVehicle);
68   }
69 }
```

## Star Wars Vehicle Sales

TIE Fighter  
Quantity: 1  
Vehicle: TIE Fighter  
Movie: Return of the Jedi  
Price: 55000  
Total: 55000

56 qtyEff = effect(() => console.log('quantity:', this.quantity()));

Elements Console Performance insights Recorder Sources Network Performance Memory Application Security Lighthouse

Application bundle generation complete. [3.806 seconds]

Page reload sent client(s).

quantity: 1

vehicle: undefined

Angular is running in development mode.

vehicle: {"id":2,"name":"AT-AT","price":10050}

vehicle: {"id":3,"name":"TIE Fighter","price":55000}

Chrome is moving towards a new experience that allows users to choose to browse without third-party cookies.

effect - Angular

angular.dev/api/core/effect

Share your experience with Angular in The State of JavaScript 2024 survey

@angular/core

effect Function Developer Preview

Registers an "effect" that will be scheduled & executed whenever the signals that it reads changes.

API Description

```
function effect(
  effectFn: (onCleanup: EffectCleanupRegisterFn) => void,
  options?: CreateEffectOptions | undefined
): EffectRef;
```

effect EffectRef

Registers an "effect" that will be scheduled & executed whenever the signals that it reads changes.

Angular has two different kinds of effect: component effects and root effects. Component effects are created when `effect()` is called from a component, directive, or within a service of a component/directive. Root effects are created when `effect()` is called from outside the component tree, such as in a root service, or when the `forceRoot` option is provided.

The two effect types differ in their timing. Component effects run as a component lifecycle event during Angular's synchronization (change detection) process, and can safely read input signals or create/destroy views that depend on component state. Root effects run as microtasks and have no connection to the component tree or change detection.

`effect()` must be run in injection context, unless the `injector` option is manually specified.



effect • Angular

main.ts - effect() Guidance (src)

angular.dev/api/core/effect

Share your experience with Angular in The State of JavaScript 2024 survey

@angular/core

v19

CM K

Docs

Tutorials

Playground

Reference

@defer

@defer

@for

@for

@if

@if

@let

@switch

@switch

AbstractType

AfterContentChecked

AfterContentInit

afterNextRender

afterRender

afterRenderEffect

AfterRenderOptions

AfterRenderPhase

AfterRenderRef

AfterViewChecked

AfterViewInit

ANIMATION\_MODULE\_TYPE

@angular/core

effect

Function Developer Preview

Registers an "effect" that will be scheduled & executed whenever the signals that it reads changes.

API Description

```
function effect(
  effectFn: (onCleanup: EffectCleanupRegisterFn) => void,
  options?: CreateEffectOptions | undefined
): EffectRef;
```

effect

EffectRef

Registers an "effect" that will be scheduled & executed whenever the signals that it reads changes.

Angular has two different kinds of effect: component effects and root effects. Component effects are created when `effect()` is called from a component, directive, or within a service of a component/directive. Root effects are created when `effect()` is called from outside the component tree, such as in a root service, or when the `forceRoot` option is provided.

The two effect types differ in their timing. Component effects run as a component lifecycle event during Angular's synchronization (change detection) process, and can safely read input signals or create/destroy views that depend on component state. Root effects run as microtasks and have no connection to the component tree or change detection.

`effect()` must be run in injection context, unless the `injector` option is manually specified.

`@param effectFn: (onCleanup: EffectCleanupRegisterFn) => void`

`@param options: CreateEffectOptions | undefined`

`@returns EffectRef`

CreateEffectOptions • Angular

main.ts - effect() Guidance (src)

angular.dev/api/core/CreateEffectOptions

Share your experience with Angular in The State of JavaScript 2024 survey

@angular/core

v19

CM K

Docs

Tutorials

Playground

Reference

@defer

@defer

@for

@for

@if

@if

@let

@switch

@switch

AbstractType

AfterContentChecked

AfterContentInit

afterNextRender

afterRender

afterRenderEffect

AfterRenderOptions

AfterRenderPhase

AfterRenderRef

AfterViewChecked

AfterViewInit

ANIMATION\_MODULE\_TYPE

@angular/core

CreateEffectOptions

Interface Developer Preview

Options passed to the `effect` function.

API

```
interface CreateEffectOptions {
  injector?: Injector | undefined;
  manualCleanup?: boolean | undefined;
  forceRoot?: true | undefined;
  allowSignalWrites?: boolean | undefined;
  debugName?: string | undefined;
}
```

injector

Injector | undefined

The `Injector` in which to create the effect.

If this is not provided, the current `injection context` will be used instead (via `inject`).

manualCleanup

boolean | undefined

Whether the `effect` should require manual cleanup.

If this is `false` (the default) the effect will automatically register itself to be cleaned up with the current `DestroyRef`.

forceRoot

true | undefined

Always create a root effect (which is scheduled as a microtask) regardless of whether `effect` is called within a component.

allowSignalWrites

boolean | undefined

`@deprecated`

stackblitz.com/~edit/effect-guidance-deborahk-n7sogg?file=src/main.ts

effect() Guidance (start) (forked)

Share

TS main.ts

```
quantity = linkedSignal({
  source: this.selectedVehicle,
  computation: () => 1
});

vehicles = signal<Vehicle[]>([
  { id: 1, name: 'Sand Crawler', price: 22050 },
  { id: 2, name: 'AT-AT', price: 10050 },
  { id: 3, name: 'TIE Fighter', price: 55000 }
]);

// Task 1: React to changes and adjust the total and color.
total = computed(() => (this.selectedVehicle()?.price ?? 0) * this.quantity());
color = computed(() => this.total() > 50000 ? 'green' : 'blue');

// Task 2: Reset the quantity when the vehicle changes

// Task 3: Retrieve the movies for the selected vehicle
http = inject(HttpClient);
movieResource = rxResource({
  request: this.selectedVehicle,
  loader: () => this.http.get<Film>(`${this.url}/${this.selectedVehicle()?.id}`)
});
movie = computed(() => this.movieResource.value());

// Task 4: Log out signals when they change
qtyEff = effect(() => console.log('quantity:', this.quantity()));
vehEff = effect(() => console.log('vehicle:', JSON.stringify(this.selectedVehicle())));
```

### Star Wars Vehicle Sales

TIE Fighter

Quantity: 1

Vehicle: TIE Fighter

Movie: Return of the Jedi

Price: \$5000

Total: \$5000

Always use effect as a last resort

stackblitz.com/~edit/effect-guidance-deborahk-n7sogg?file=src/main.ts

effect() Guidance (start) (forked)

Share

TS main.ts

```
quantity = linkedSignal({
  source: this.selectedVehicle,
  computation: () => 1
});

vehicles = signal<Vehicle[]>([
  { id: 1, name: 'Sand Crawler', price: 22050 },
  { id: 2, name: 'AT-AT', price: 10050 },
  { id: 3, name: 'TIE Fighter', price: 55000 }
]);

// Task 1: React to changes and adjust the total and color
total = computed(() => (this.selectedVehicle()?.price ?? 0) * this.quantity());
color = computed(() => this.total() > 50000 ? 'green' : 'blue');

// Task 2: Reset the quantity when the vehicle changes

// Task 3: Retrieve the movies for the selected vehicle
http = inject(HttpClient);
movieResource = rxResource({
  request: this.selectedVehicle,
  loader: () => this.http.get<Film>(`${this.url}/${this.selectedVehicle()?.id}`)
});
movie = computed(() => this.movieResource.value());

// Task 4: Log out signals when they change
qtyEff = effect(() => console.log('quantity:', this.quantity()));
vehEff = effect(() => console.log('vehicle:', JSON.stringify(this.selectedVehicle())));
```

```
graph TD
    Q1[React to signal changes?] -- No --> A1[Use a regular function]
    Q1 -- Yes --> Q2[Set a readonly signal?]
    Q2 -- Yes --> A2[Use computed()]
    Q2 -- No --> Q3[Set a writeable signal?]
    Q3 -- Yes --> A3[Use linkedSignal()]
    Q3 -- No --> Q4[Perform an async operation?]
    Q4 -- Yes --> A4[Use resource() or rxResource()]
    Q4 -- No --> A5[Use an effect()]
```