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

Deborah Kurata
19.6K subscribers

Subscribe

408 | Share | Download | Thanks | ...

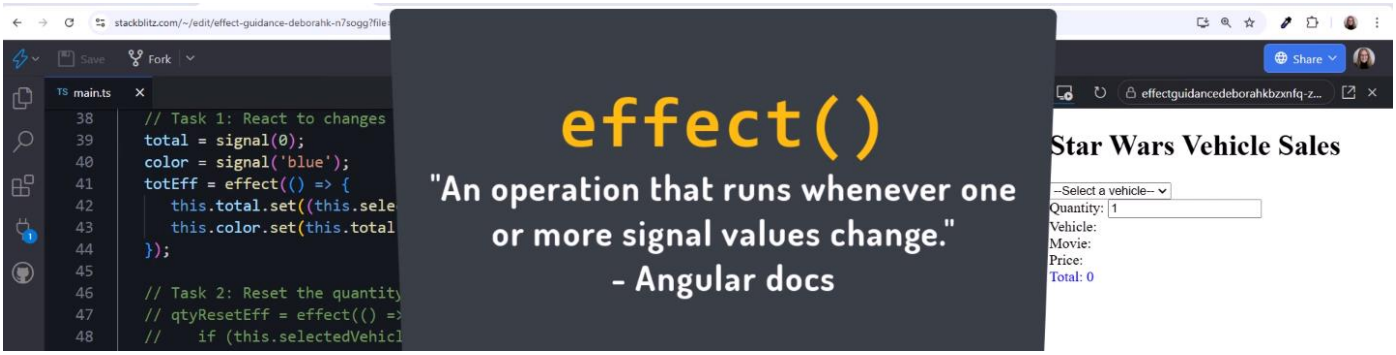
5,735 views Dec 4, 2024 #angular #effect #resource

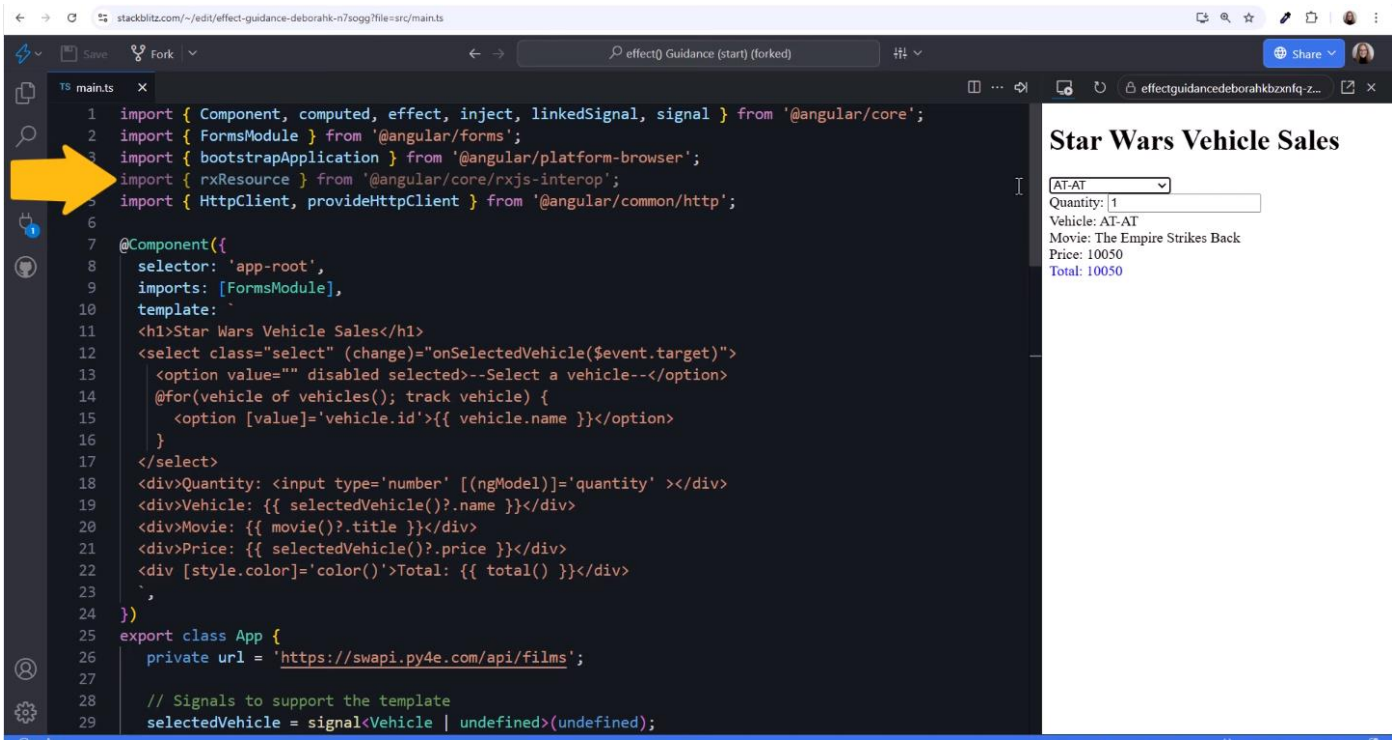
"An effect is an operation that runs whenever one or more signal values change." Does that mean we should use it anytime we want to react to changes in a signal?

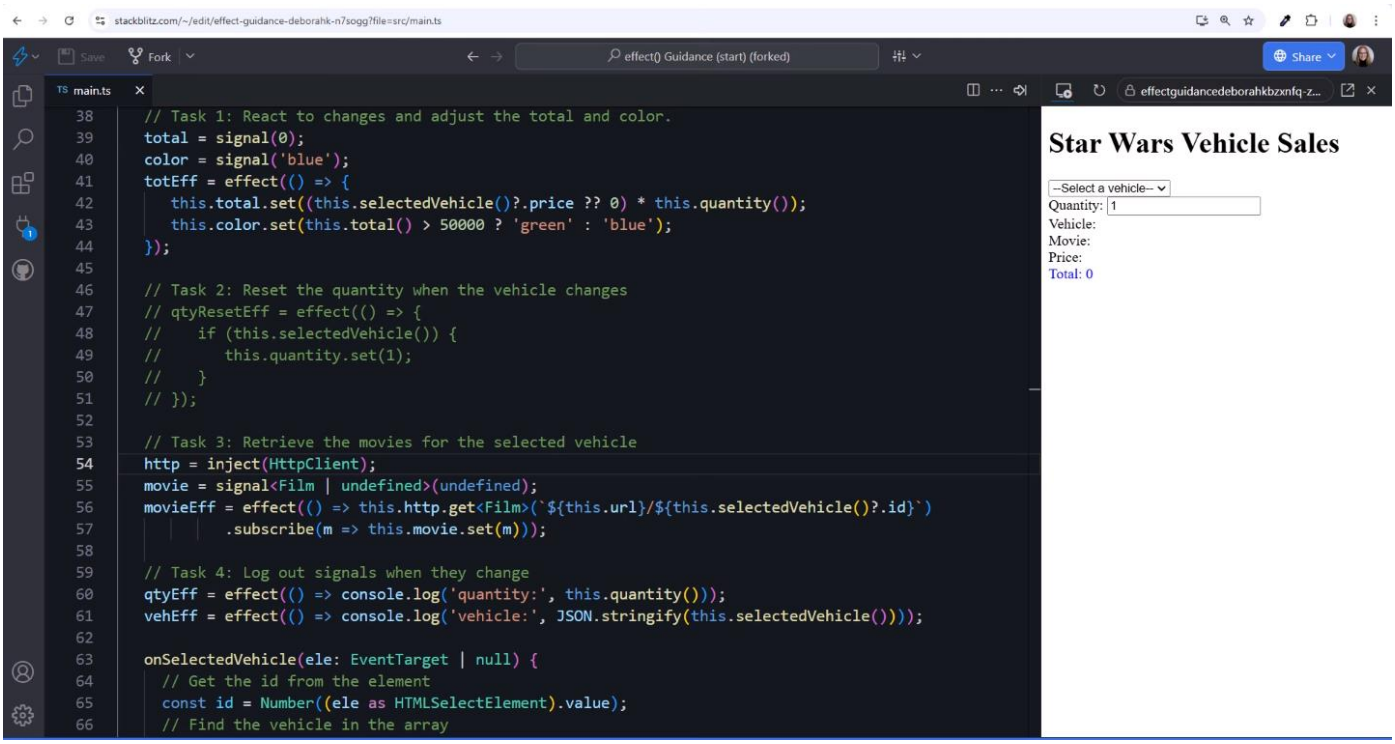
Angular v19 includes changes to effect, and introduces several additional signal features. Does that change the guidance?

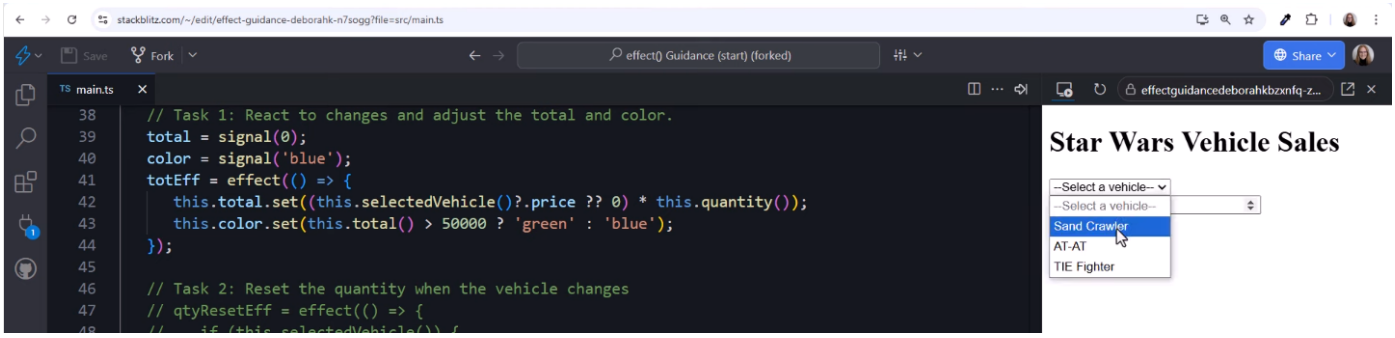
In this video, we walk through several scenarios, examining an effect() based approach and an alternate approach. Then we look at the version 19 changes to the effect() function and summarize suggested guidelines.

Content



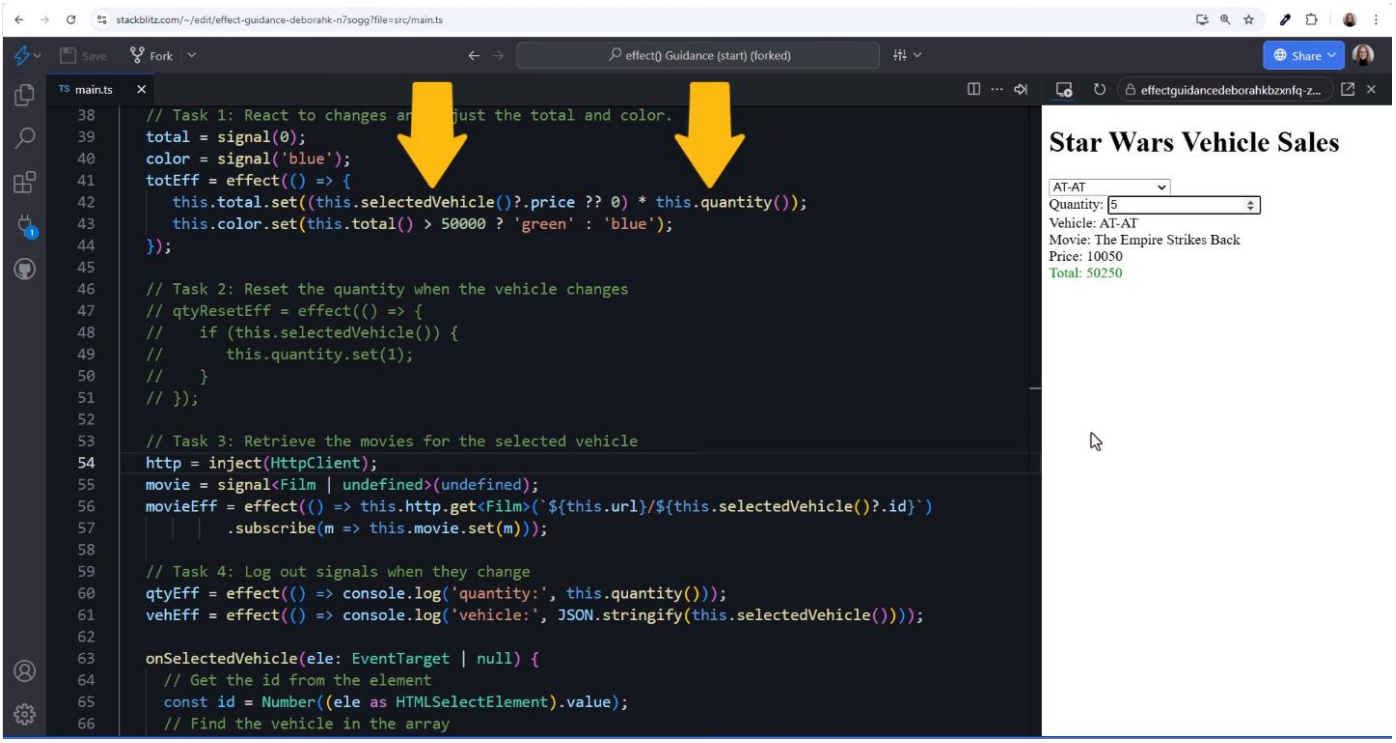


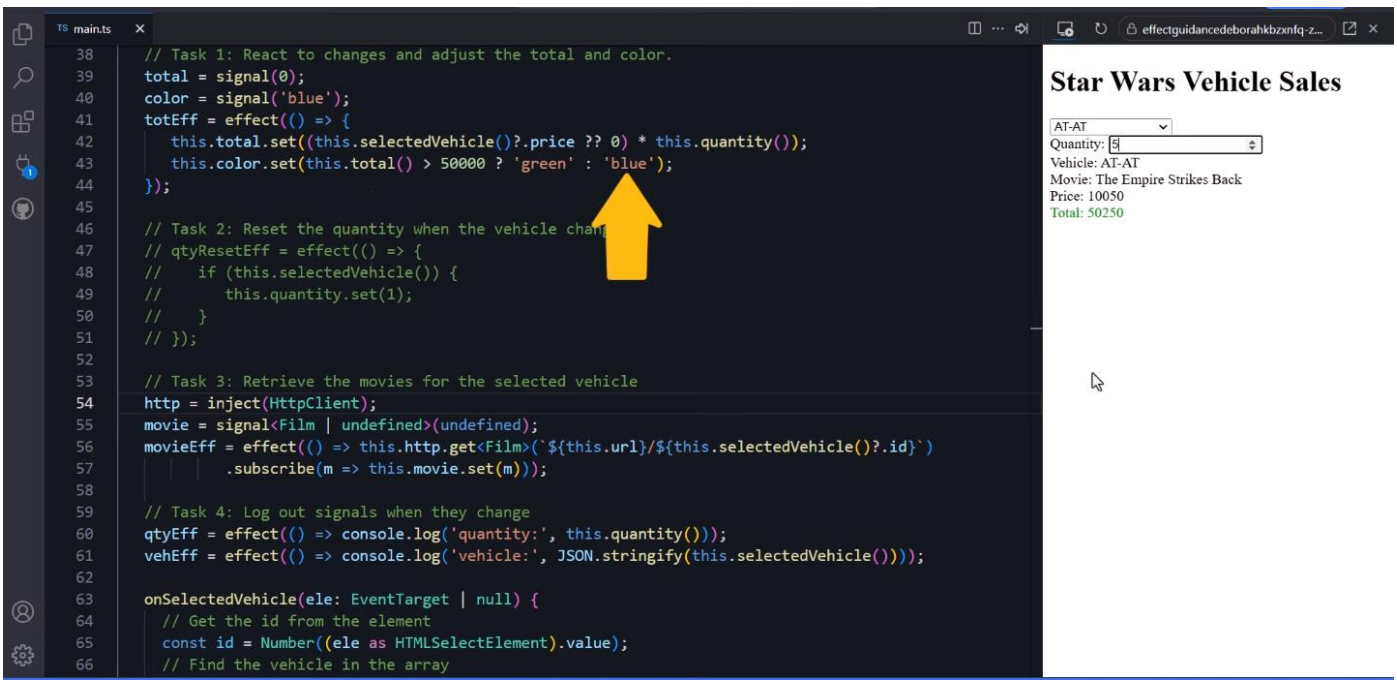

stackblitz.com/~/edit/effect-guidance-deborahk-n7sogg?file=src/main.ts
Save
Fork
effect() Guidance (start) (forked)
Share
main.ts
effectguidancedeborahkbzxnfq-z...
38 // Task 1: React to changes and adjust the total and color.
39 total = signal(0);
40 color = signal('blue');
41 totEff = effect(() => {
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
--Select a vehicle--
Quantity: 1
Vehicle:
Movie:
Price:
Total: 0


stackblitz.com/~/edit/effect-guidance-deborahk-n7sogg?file=src/main.ts
Save
Fork
effect() Guidance (start) (forked)
Share
main.ts
effectguidancedeborahkbzxnfq-z...
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(() => {
Star Wars Vehicle Sales
--Select a vehicle--
--Select a vehicle--
Sand Crawler
AT-AT
TIE Fighter

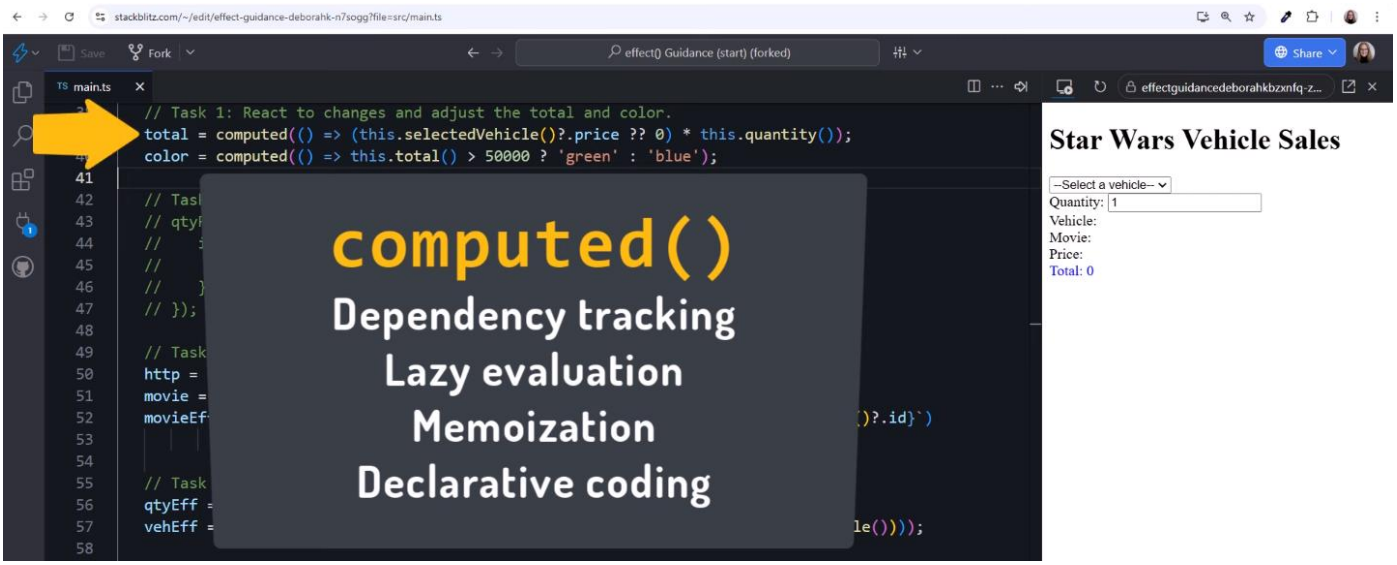

stackblitz.com/~/edit/effect-guidance-deborahk-n7sogg?file=src/main.ts
Save
Fork
effect() Guidance (start) (forked)
Share
main.ts
effectguidancedeborahkbzxnfq-z...
38 // Task 1: React to changes and adjust the total and color.
39 total = signal(0);
40 color = signal('blue');
41 totEff = effect(() => {
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
AT-AT
Quantity: 4
Vehicle: AT-AT
Movie: The Empire Strikes Back
Price: 10050
Total: 40200

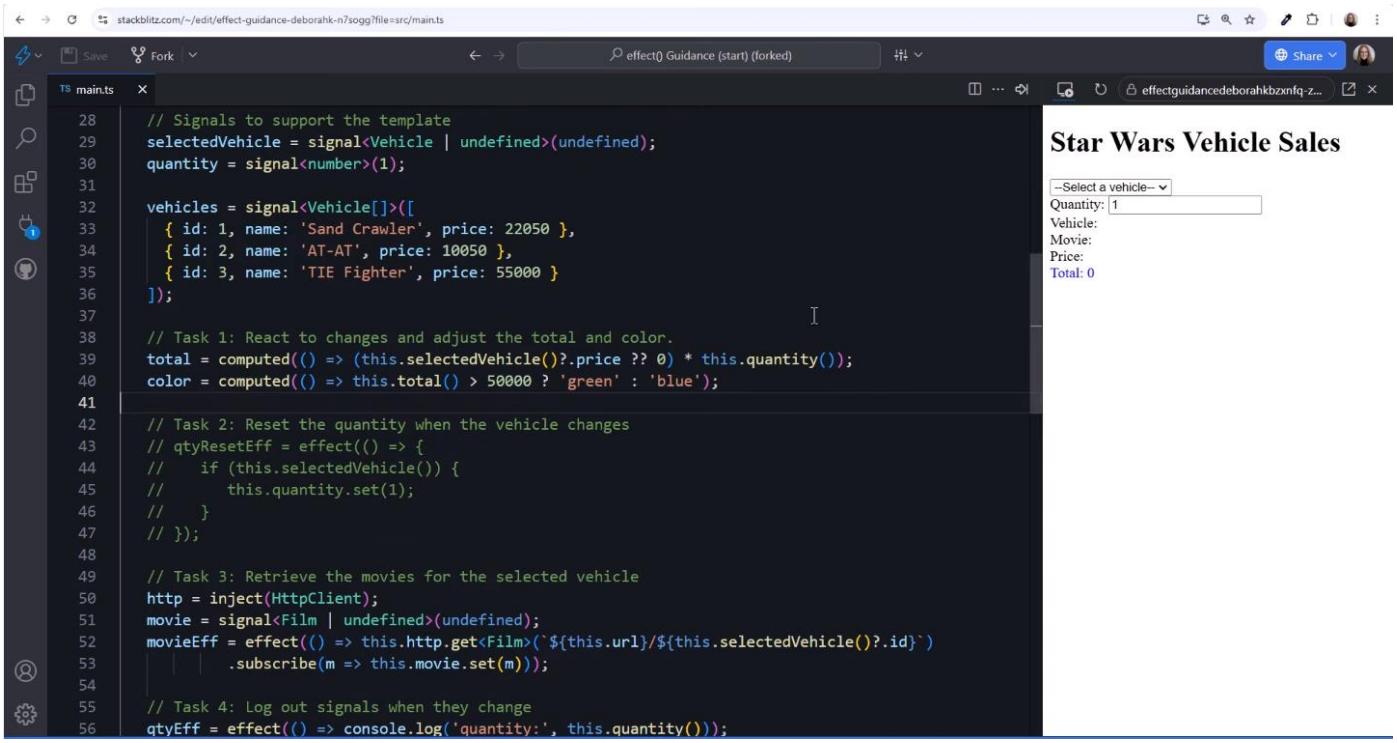

stackblitz.com/~/edit/effect-guidance-deborahk-n7sogg?file=src/main.ts
Save
Fork
effect() Guidance (start) (forked)
Share
main.ts
effectguidancedeborahkbzxnfq-z...
38 // Task 1: React to changes and adjust the total and color.
39 total = signal(0);
40 color = signal('blue');
41 totEff = effect(() => {
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
AT-AT
Quantity: 5
Vehicle: AT-AT
Movie: The Empire Strikes Back
Price: 10050
Total: 50250

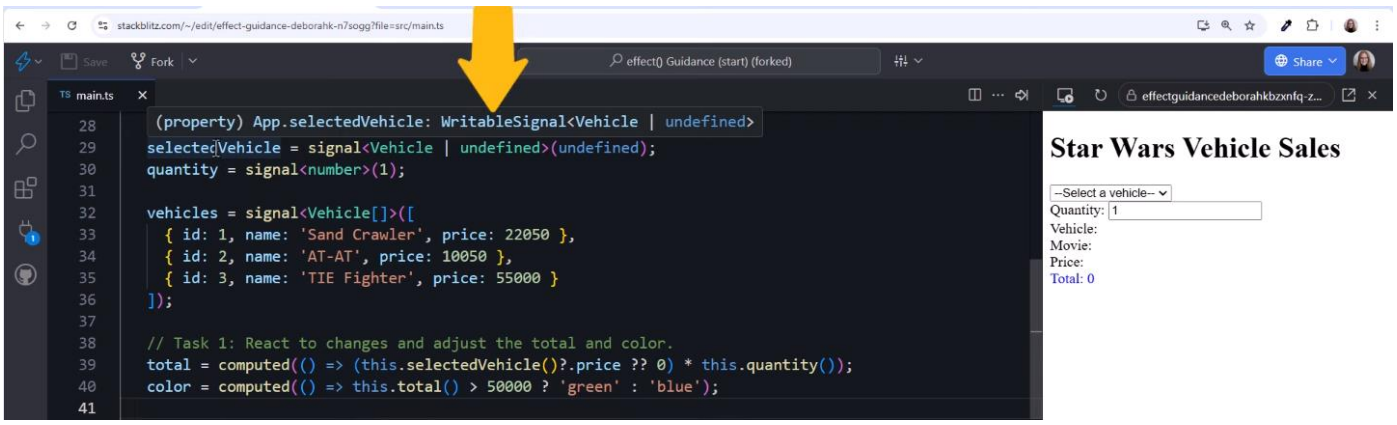


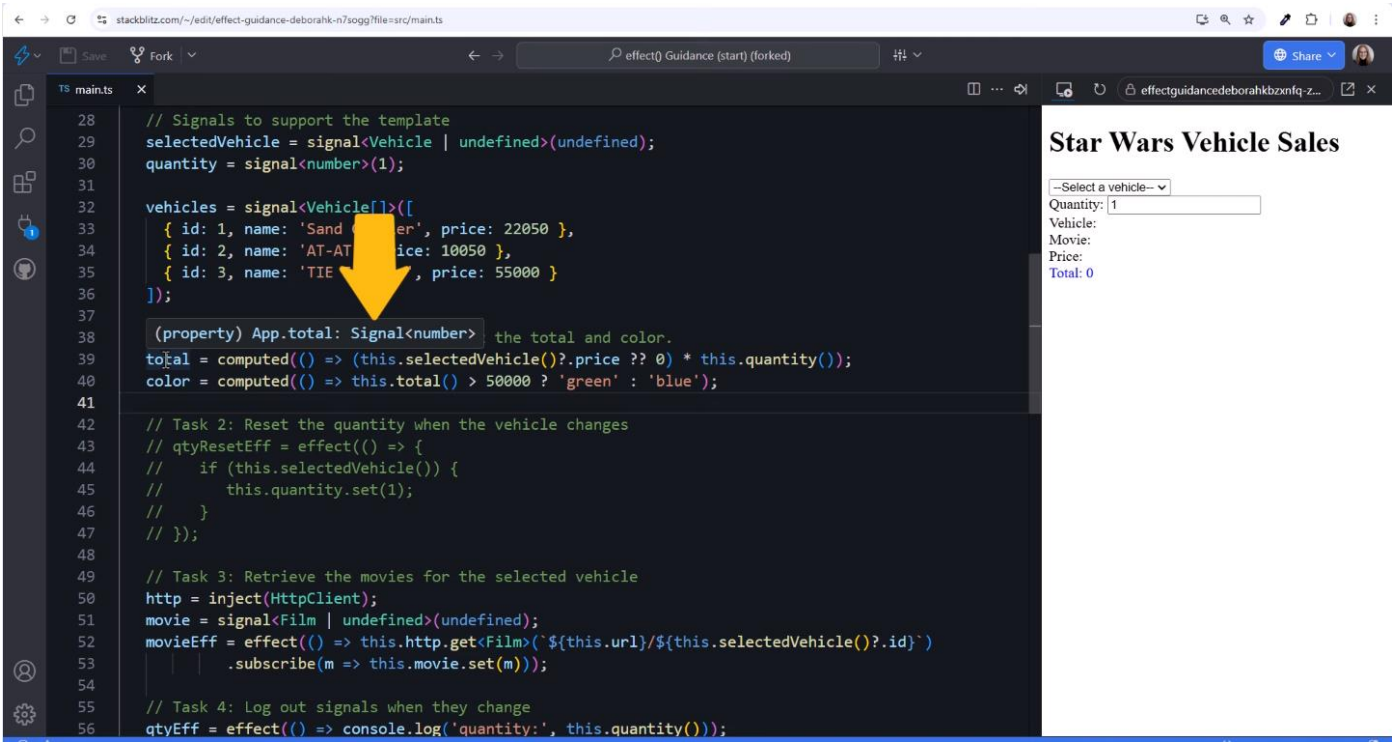


There is a much better way to accomplish this task using **computed()** signals as below

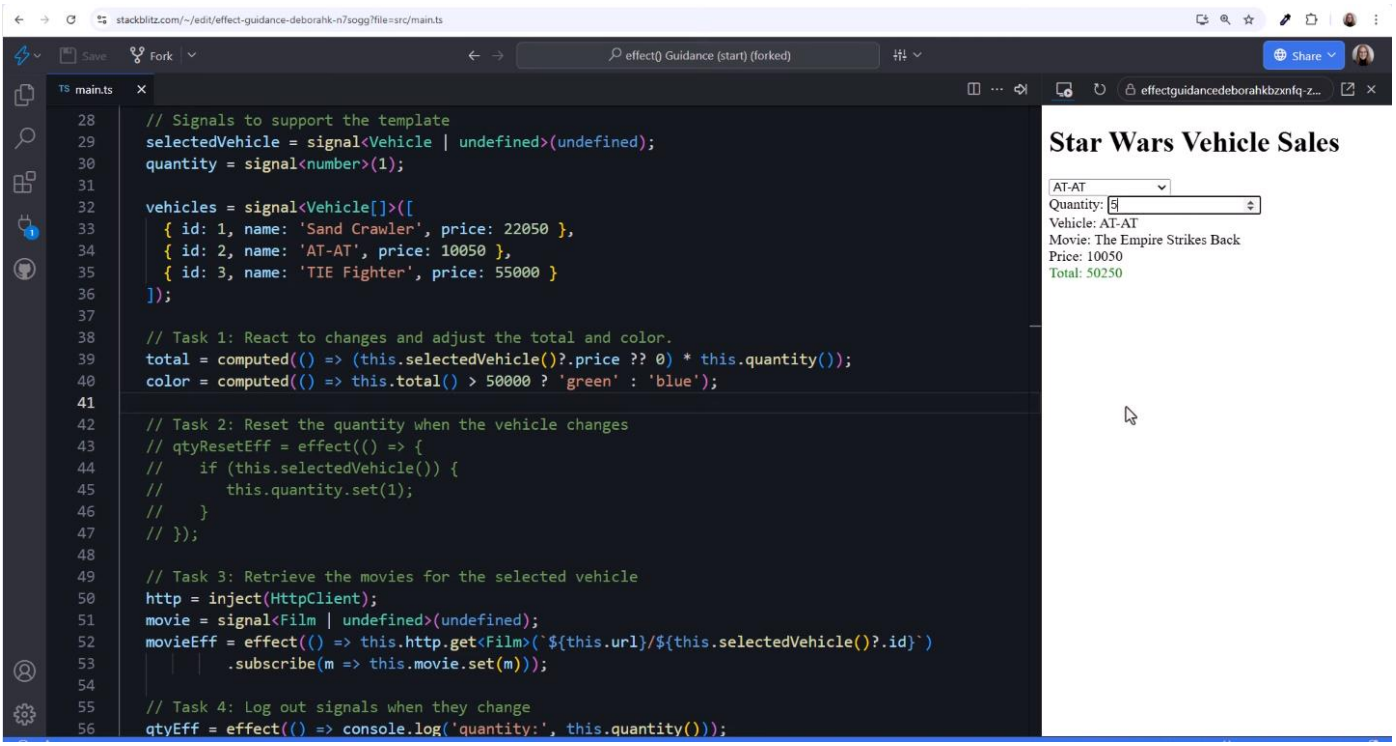

computed()
Dependency tracking
Lazy evaluation
Memoization
Declarative coding
// Task 1: React to changes and adjust the total and color.
total = computed(() => (this.selectedVehicle()?.price ?? 0) * this.quantity());
color = computed(() => this.total() > 50000 ? 'green' : 'blue');
Star Wars Vehicle Sales
--Select a vehicle--
Quantity: 1
Vehicle:
Movie:
Price:
Total: 0

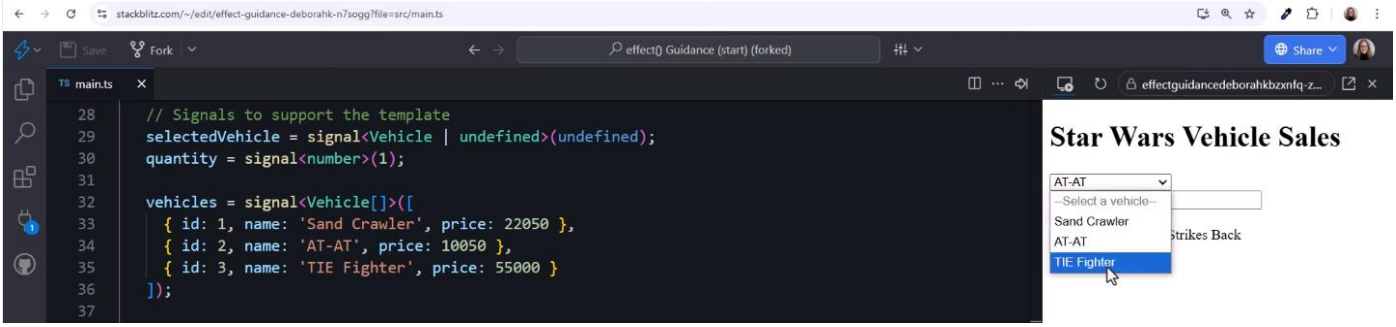

// Signals to support the template
selectedVehicle = signal<Vehicle | undefined>(undefined);
quantity = signal<number>(1);
vehicles = signal<Vehicle[]>([
{ id: 1, name: 'Sand Crawler', price: 22050 },
{ id: 2, name: 'AT-AT', price: 10050 },
{ id: 3, name: 'TIE Fighter', price: 55000 }
]);
// Task 1: 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

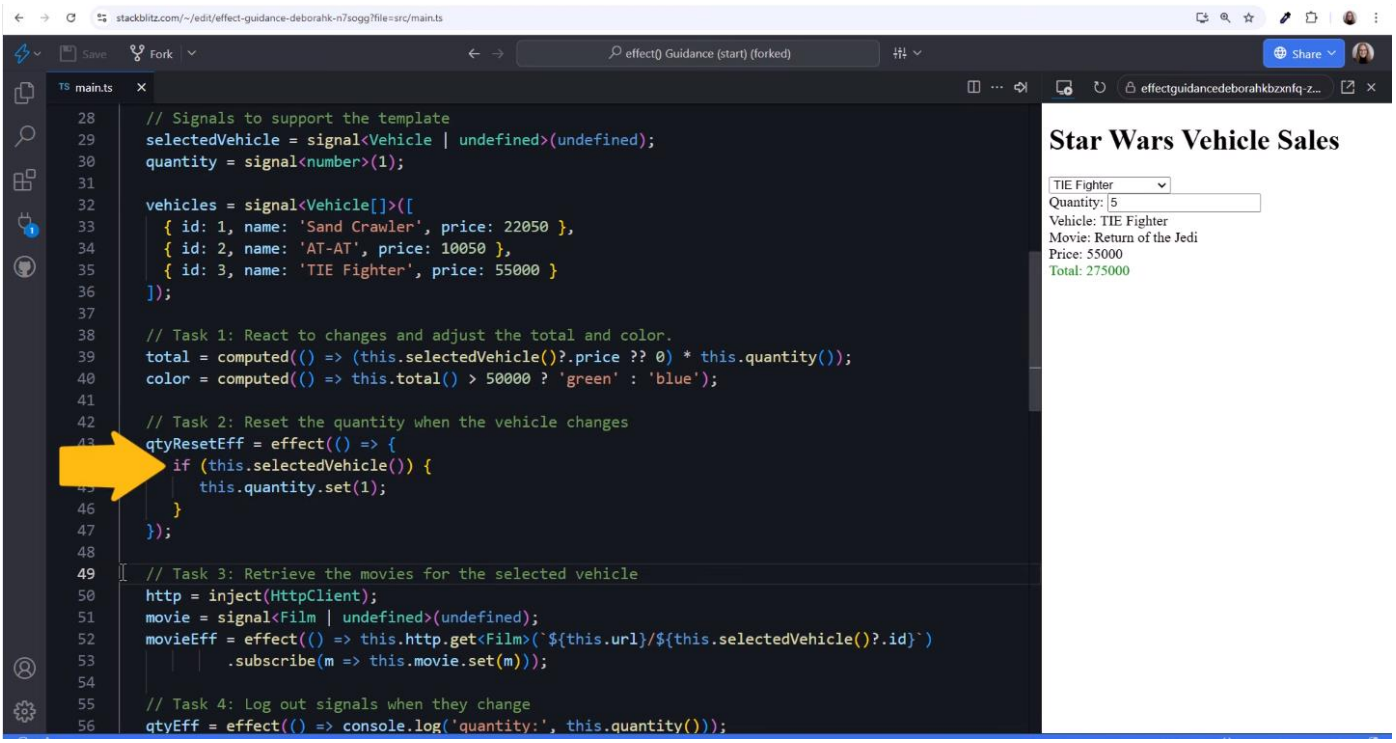

(property) App.selectedVehicle: WritableSignal<Vehicle | undefined>
selectedVehicle = signal<Vehicle | undefined>(undefined);
quantity = signal<number>(1);
vehicles = signal<Vehicle[]>([
{ id: 1, name: 'Sand Crawler', price: 22050 },
{ id: 2, name: 'AT-AT', price: 10050 },
{ id: 3, name: 'TIE Fighter', 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

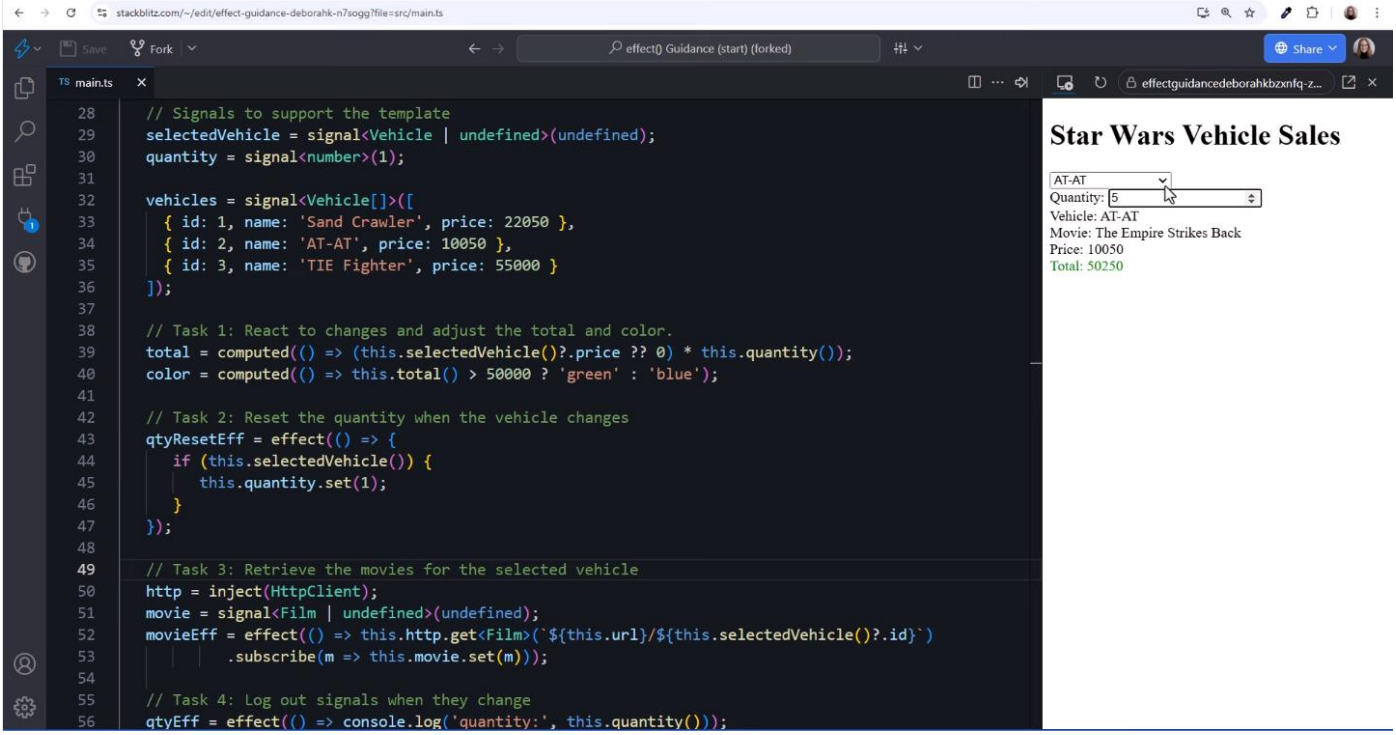


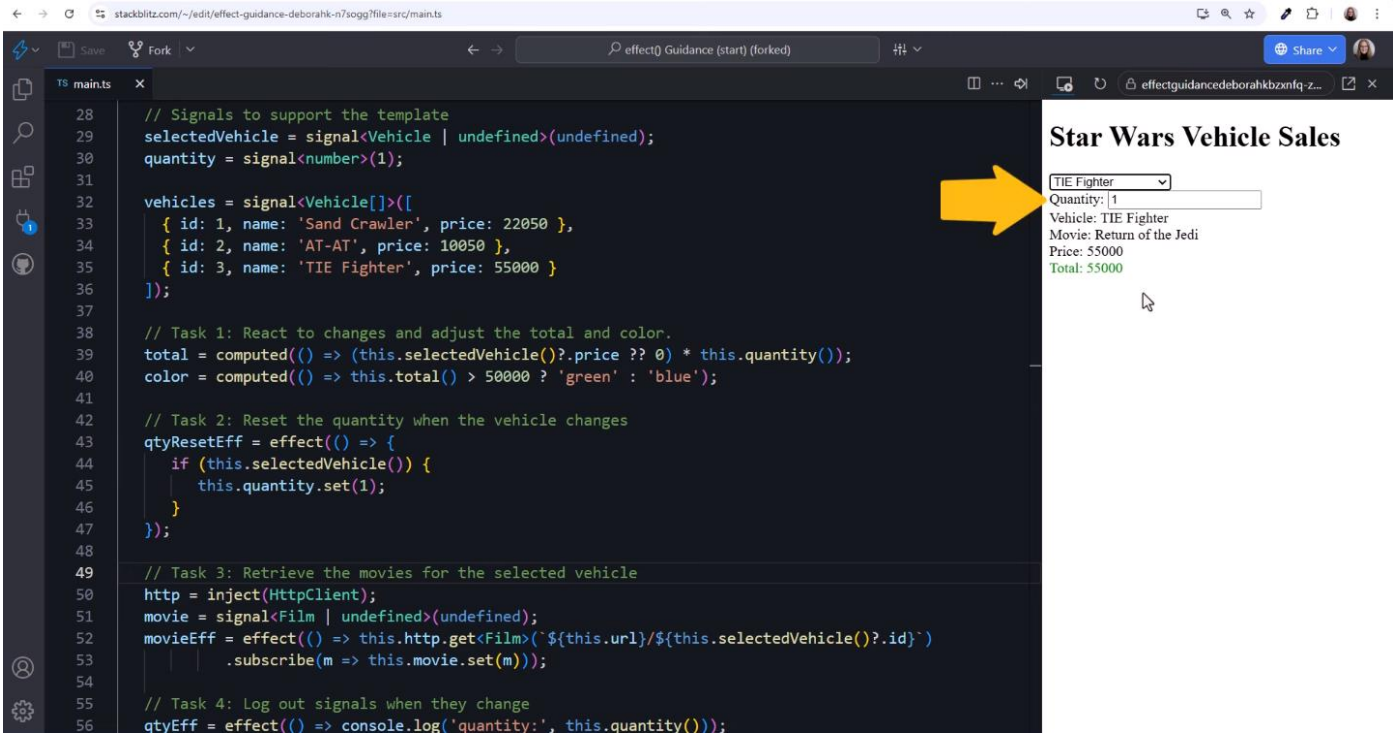
Next, let us see how to update



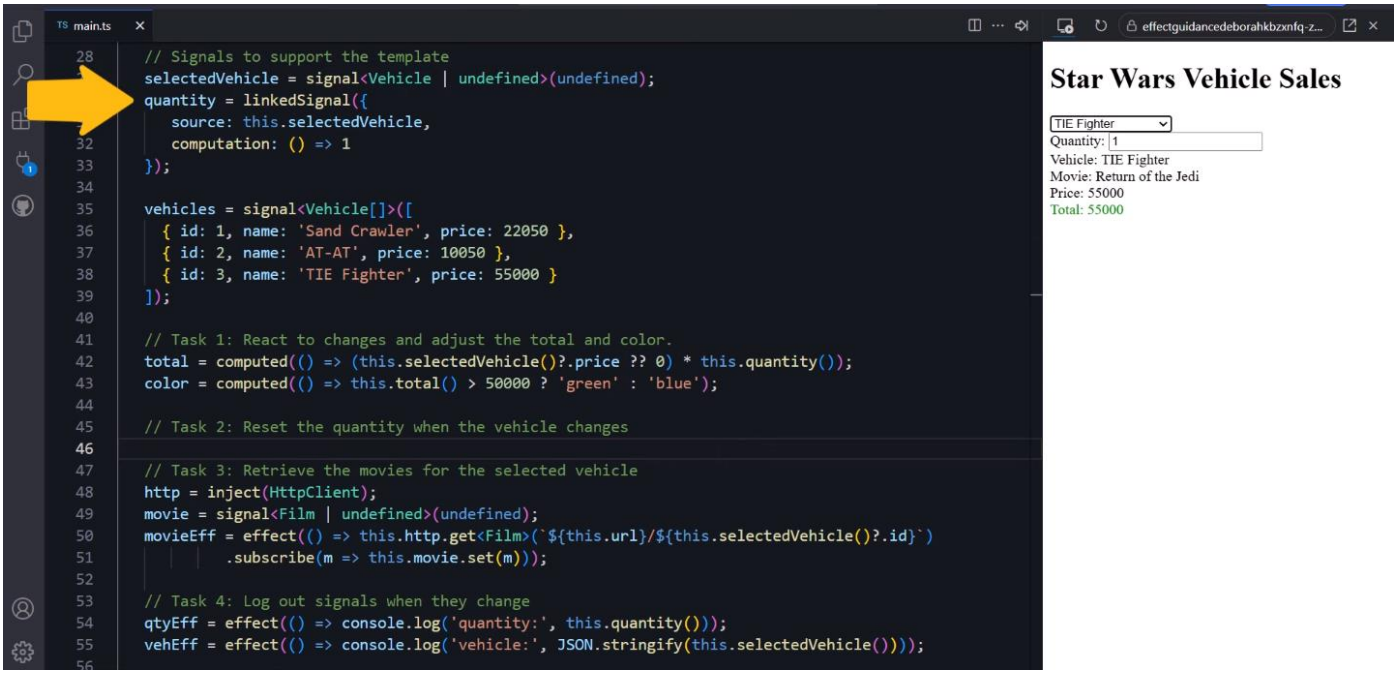


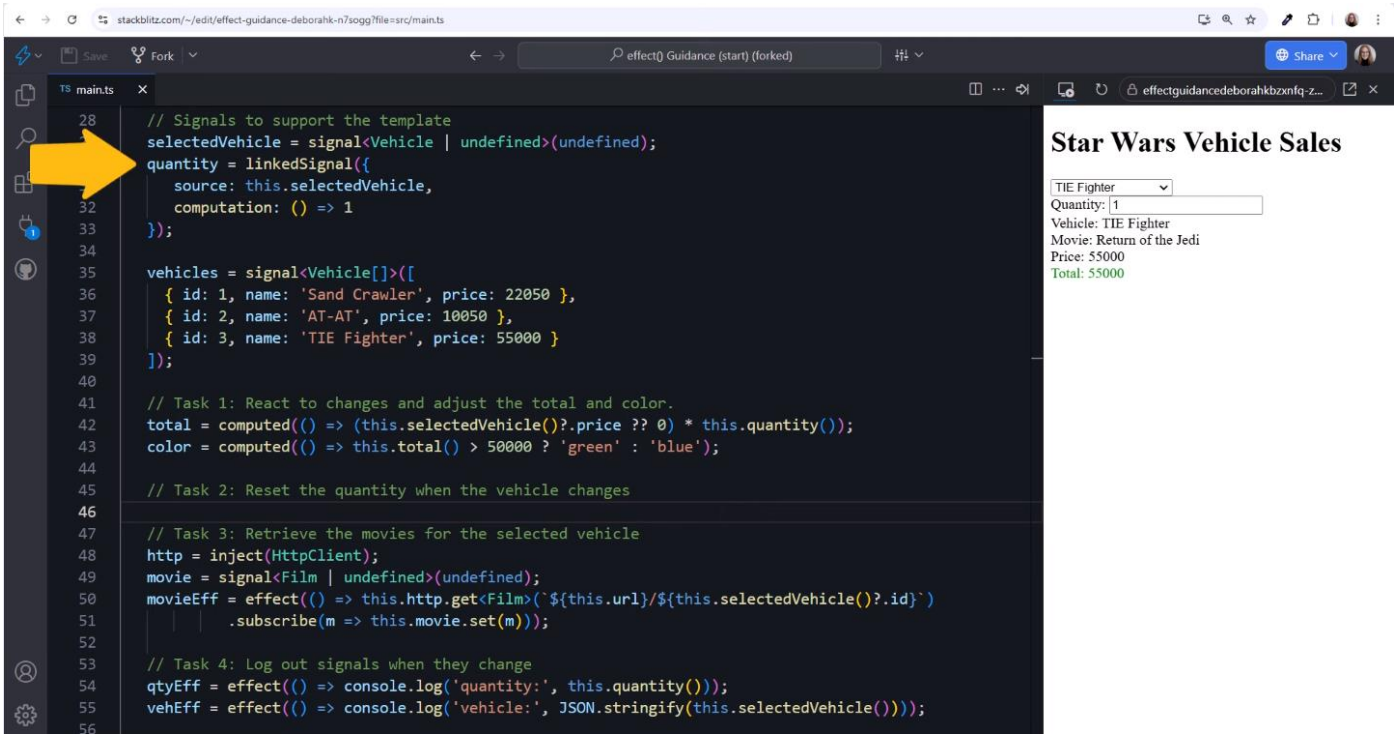

stackblitz.com/~/edit/effect-guidance-deborahk-n7sogg?file=src/main.ts
Save
Fork
effect() Guidance (start) (forked)
Share
main.ts
effectguidancedeborahkbzxnfq-z...
// Signals to support the template
selectedVehicle = signal<Vehicle | undefined>(undefined);
quantity = signal<number>(1);
vehicles = signal<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
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
Save
Fork
effect() Guidance (start) (forked)
Share
main.ts
effectguidancedeborahkbzxnfq-z...
// 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
AT-AT
Quantity: 5
Vehicle: AT-AT
Movie: The Empire Strikes Back
Price: 10050
Total: 50250

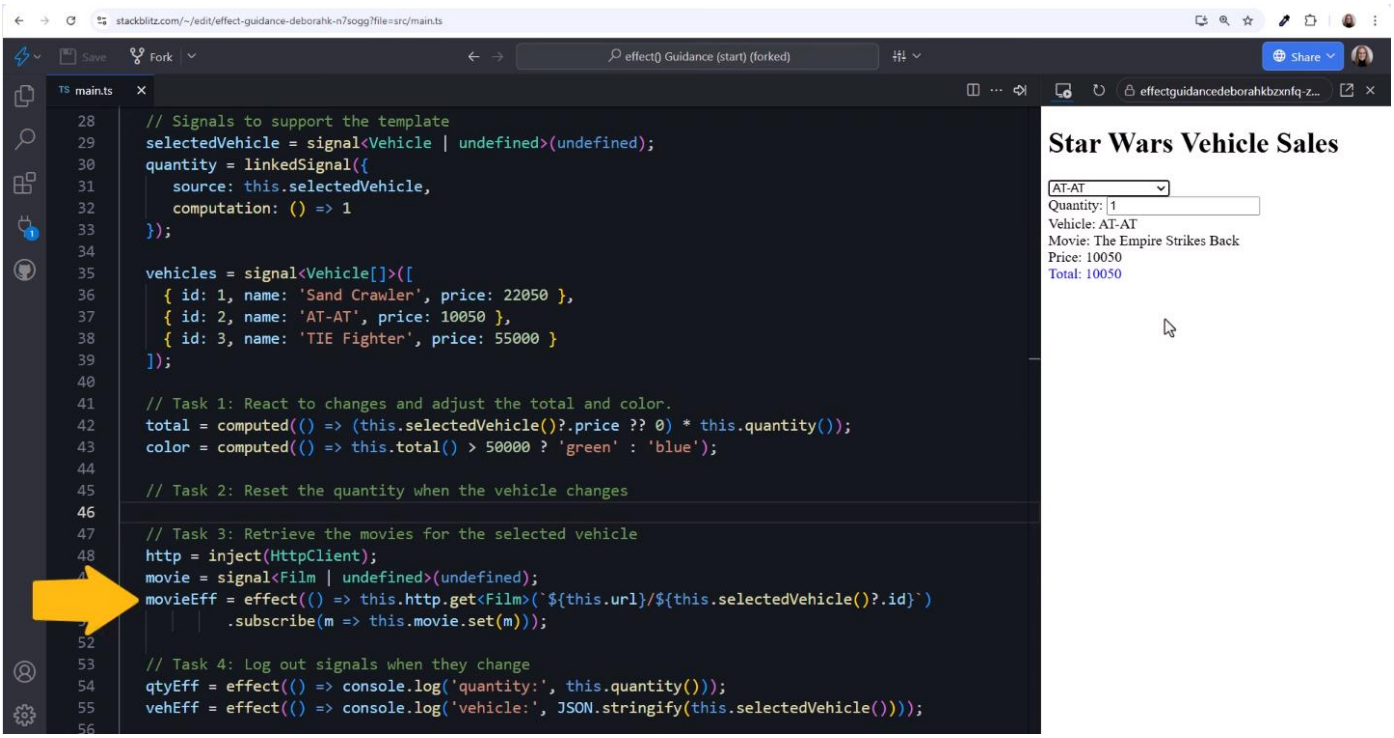


It works. But there is a better way starting in version 19 where we now have **LinkedSignal**

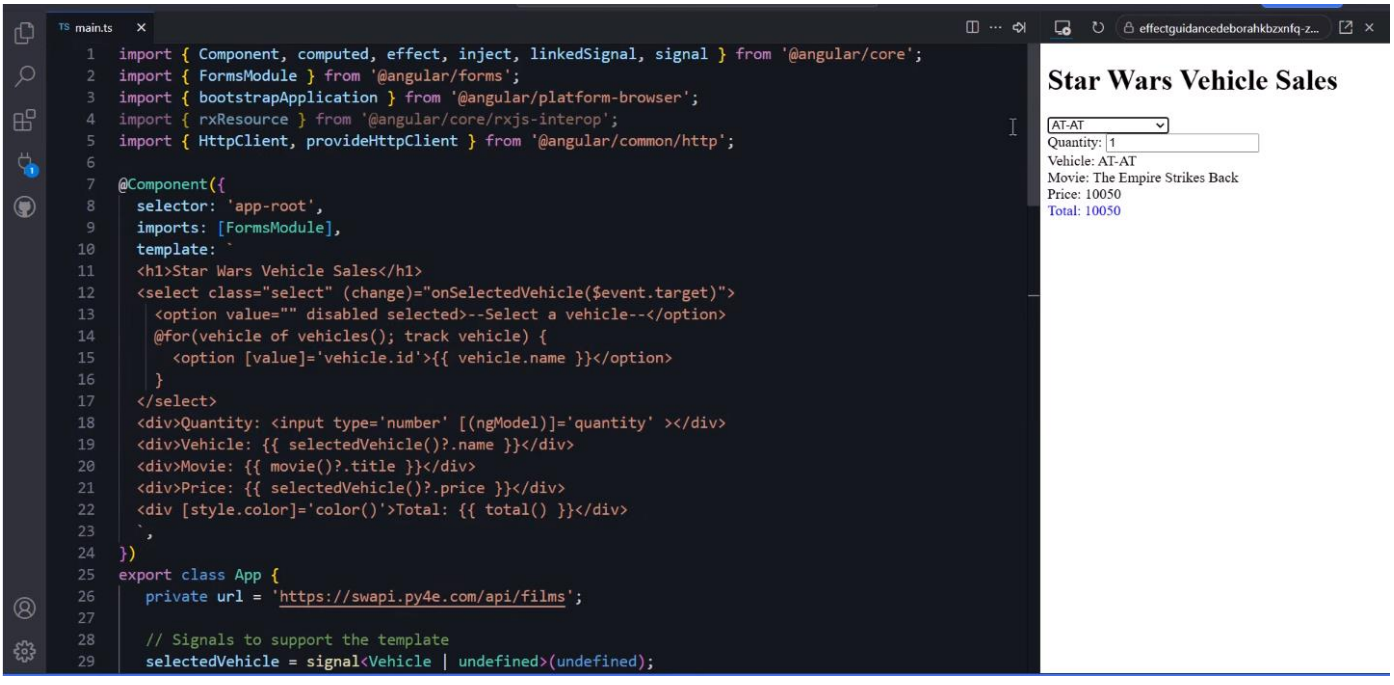


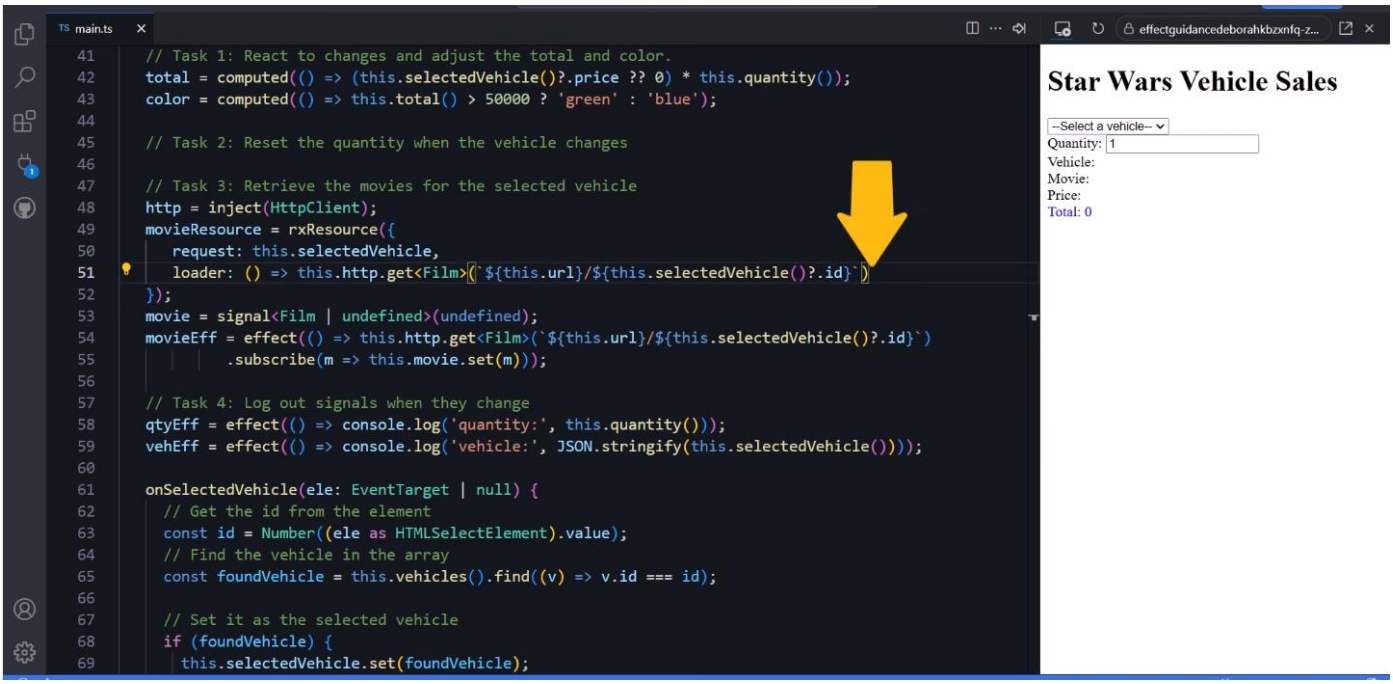


No need for an effect here anymore!

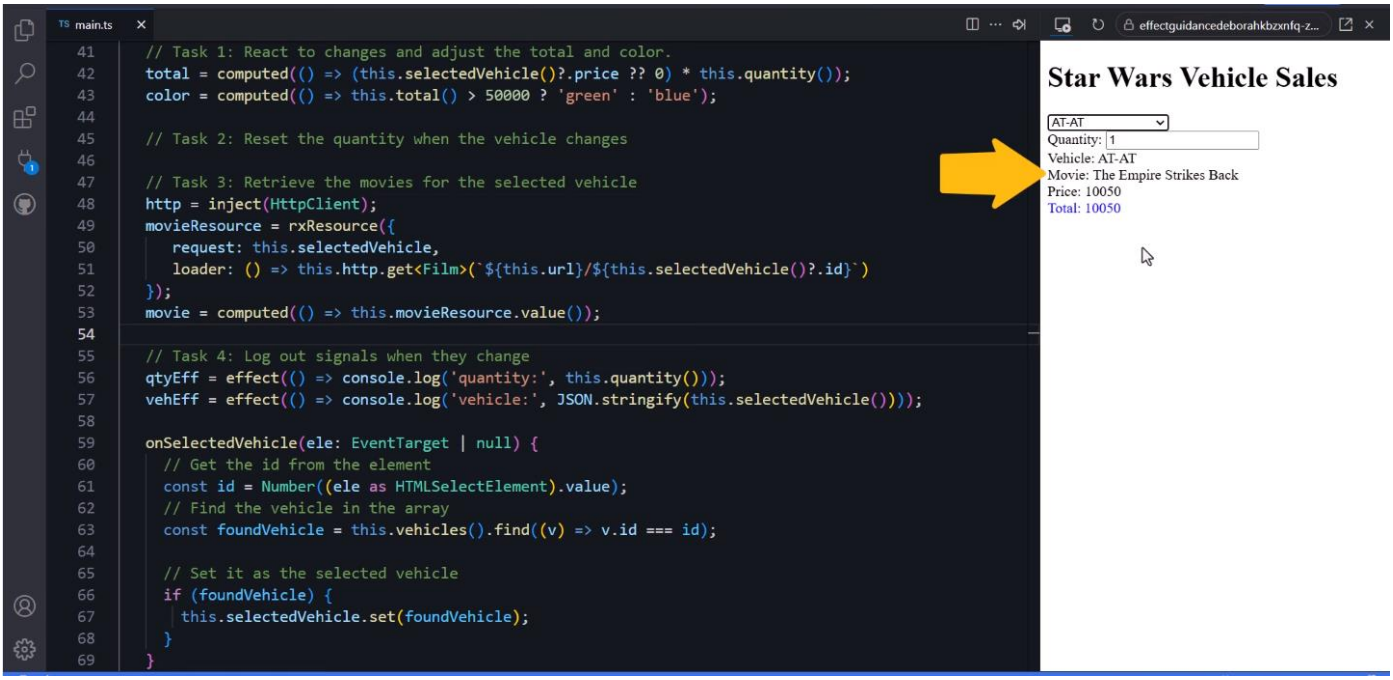


We can use the new **rxResource()** here instead of using an effect and having to manage the subscriptions.

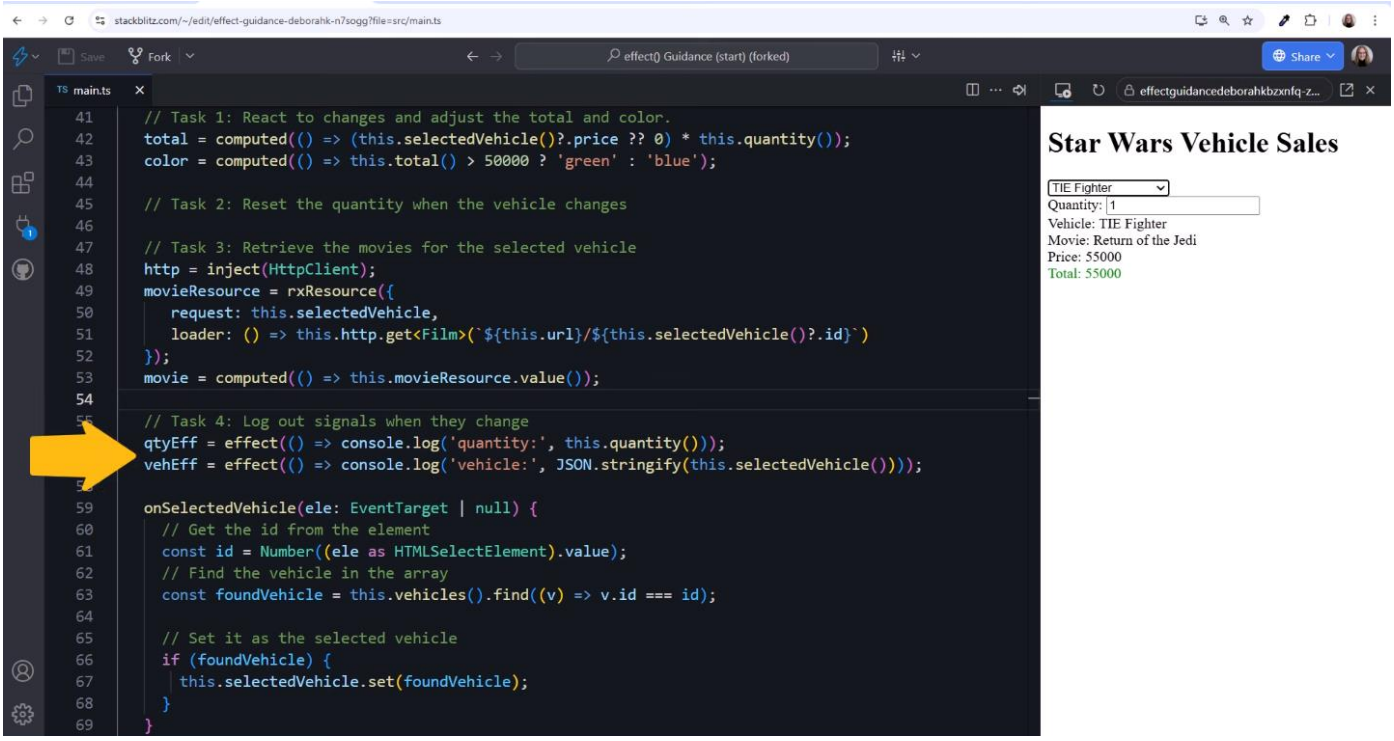


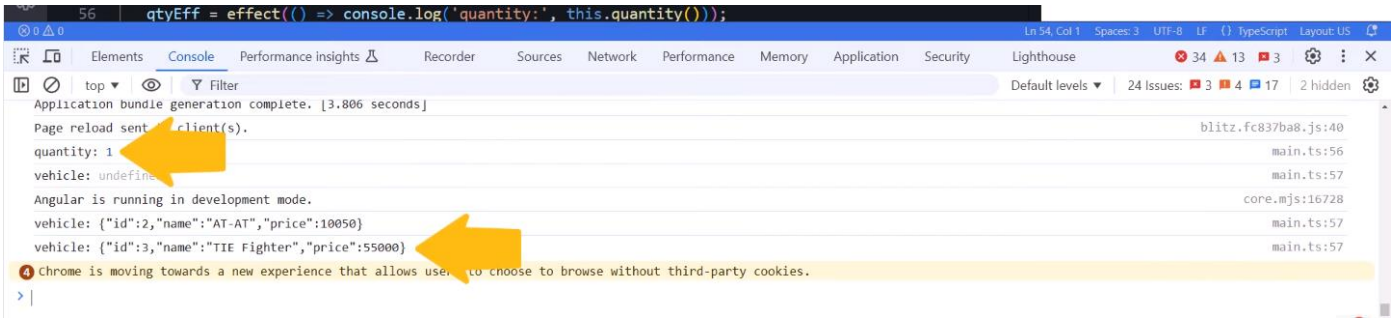


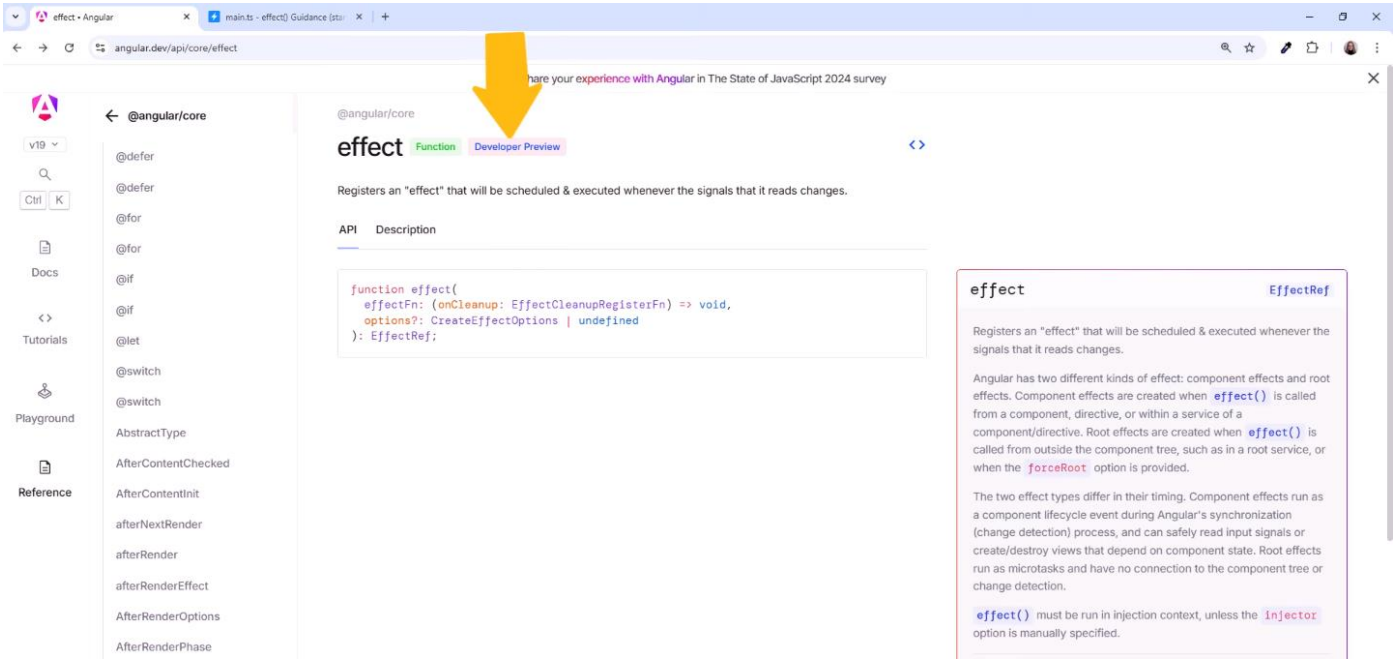
Notice that we don't have to subscribe anymore since rxResource() is managing it for us

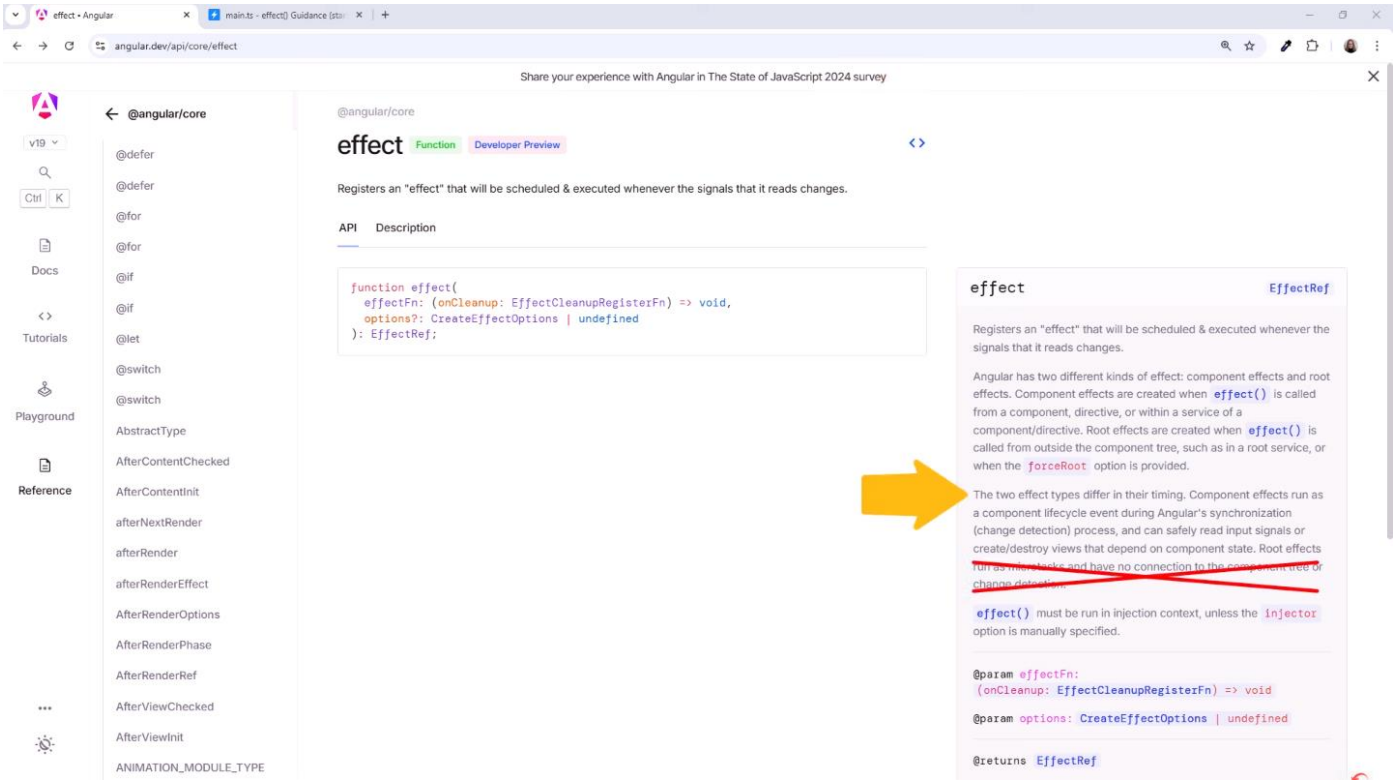


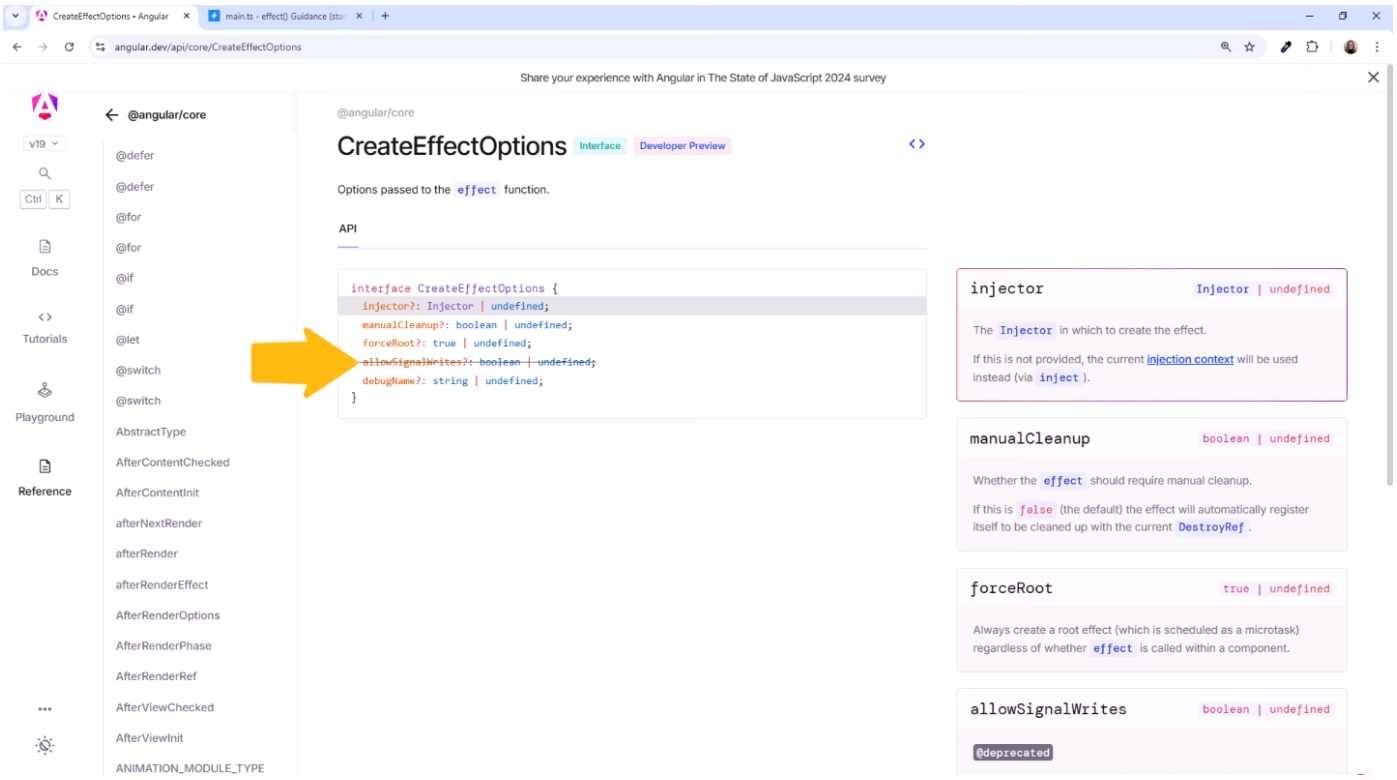
Next, let us see the effect use-case for logging out signals as they change

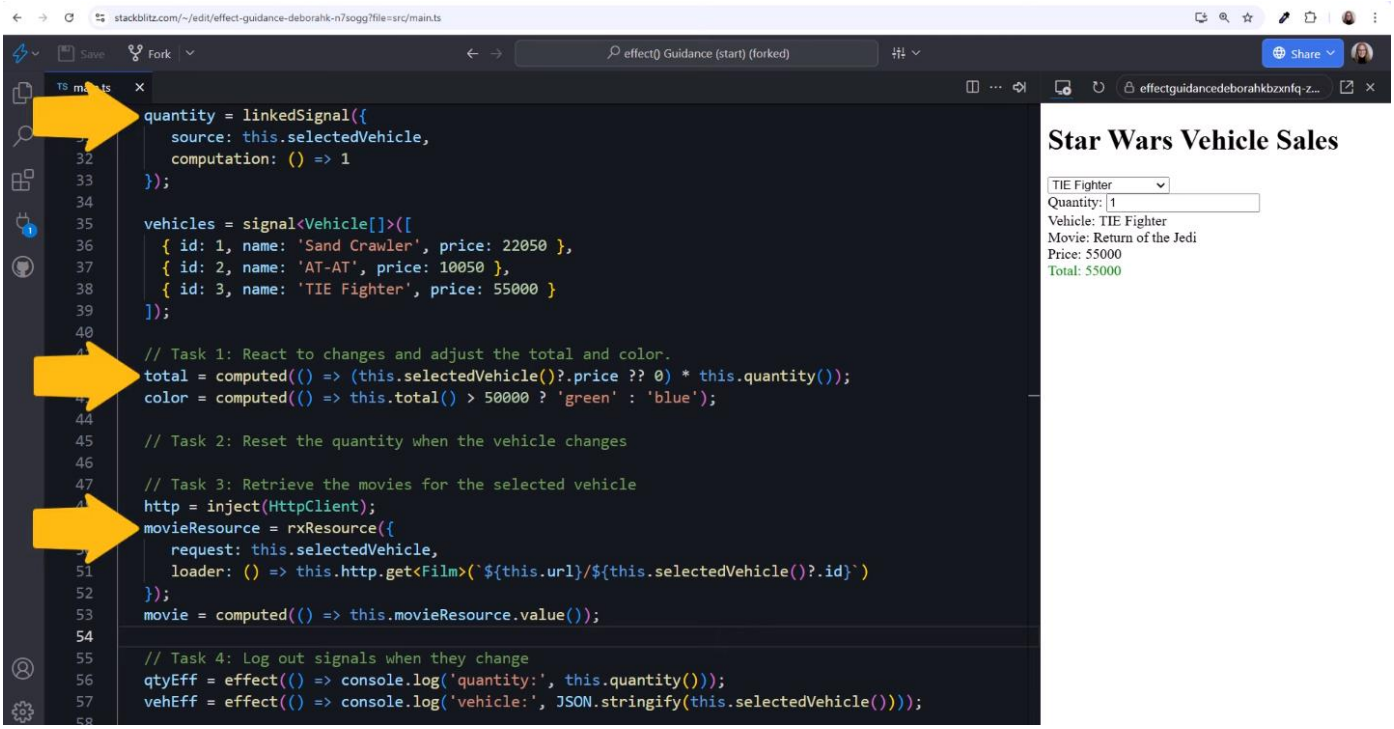

stackblitz.com/~/edit/effect-guidance-deborahk-n7sogg?file=src/main.ts
Fork
effect() Guidance (start) (forked)
Share
main.ts
effectguidancedeborahkbzxnfq-z...
// Task 1: React to changes and adjust the total and color.
total = computed(() => (this.selectedVehicle()?.price ?? 0) * this.quantity());
color = 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())));
onSelectedVehicle(ele: EventTarget | null) {
// Get the id from the element
const id = Number((ele as HTMLSelectElement).value);
// Find the vehicle in the array
const foundVehicle = this.vehicles().find((v) => v.id === id);
// Set it as the selected vehicle
if (foundVehicle) {
this.selectedVehicle.set(foundVehicle);
}
}
Star Wars Vehicle Sales
TIE Fighter
Quantity: 1
Vehicle: TIE Fighter
Movie: Return of the Jedi
Price: 55000
Total: 55000


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 to 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.


angular.dev/api/core/effect
@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 (star
angular.dev/api/core/effect
Share your experience with Angular in The State of JavaScript 2024 survey
@angular/core
v19
Ctrl K
Docs
Tutorials
Playground
Reference
@angular/core
@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 (star
angular.dev/api/core/CreateEffectOptions
Share your experience with Angular in The State of JavaScript 2024 survey
@angular/core
v19
Ctrl K
Docs
Tutorials
Playground
Reference
@angular/core
@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



Always use effect as a last resort