Angular 16: Signals

RECONCEPT_

Angular 16: Signals

Why & What

- why?
- signal()
- computed()
- effect()

Advanced

- SignalOptions
- toSignal
- toObservable

Future

- Signal Components

Why Signals?

Why a new Signal library?

`ref()` `signal()` `signal()` `useState()`

Needed something that can tie into Angular change detection better

Angular Change Detection

ZoneJS

Monkey patch all browser events

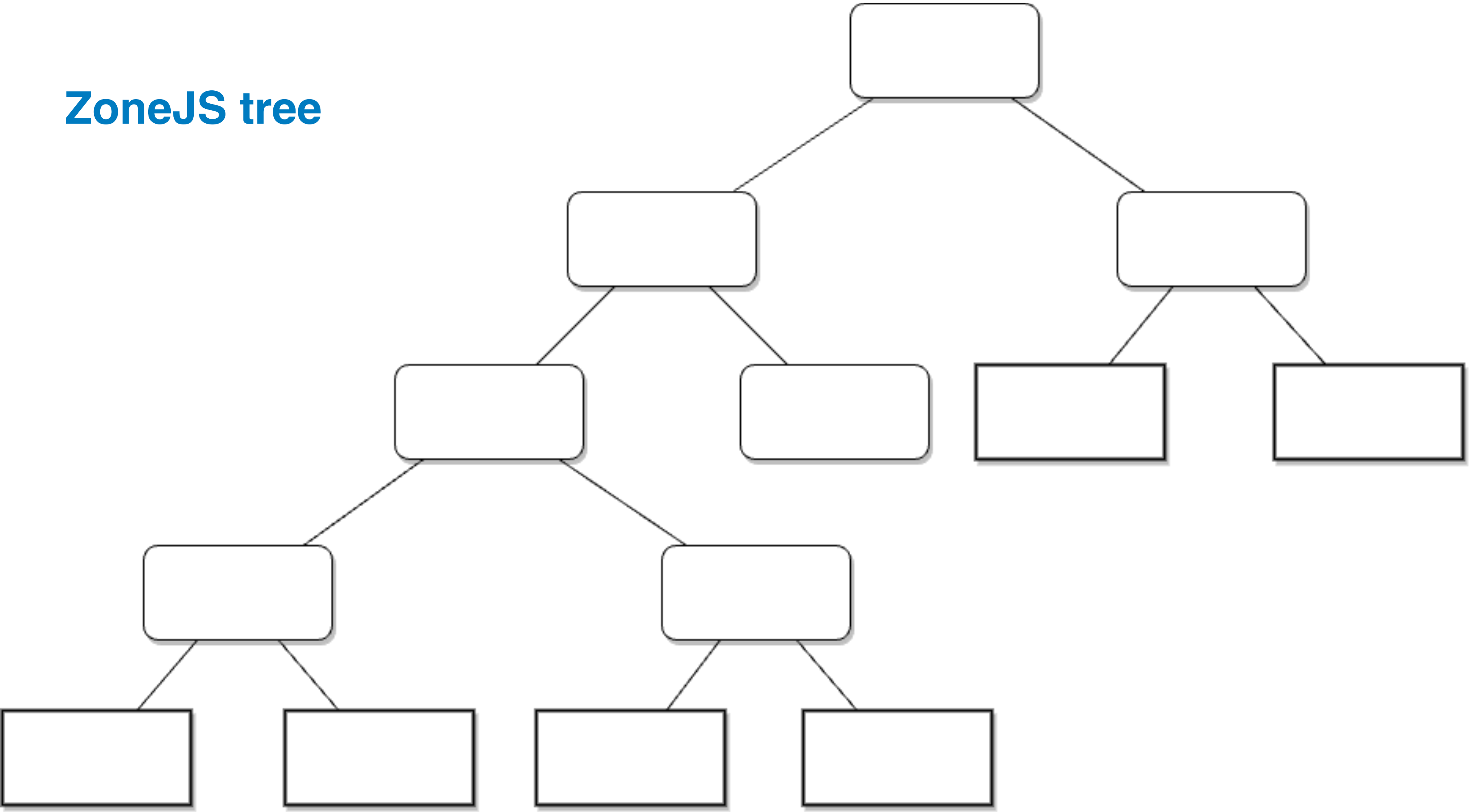
Application wide change detection

Runs top down*

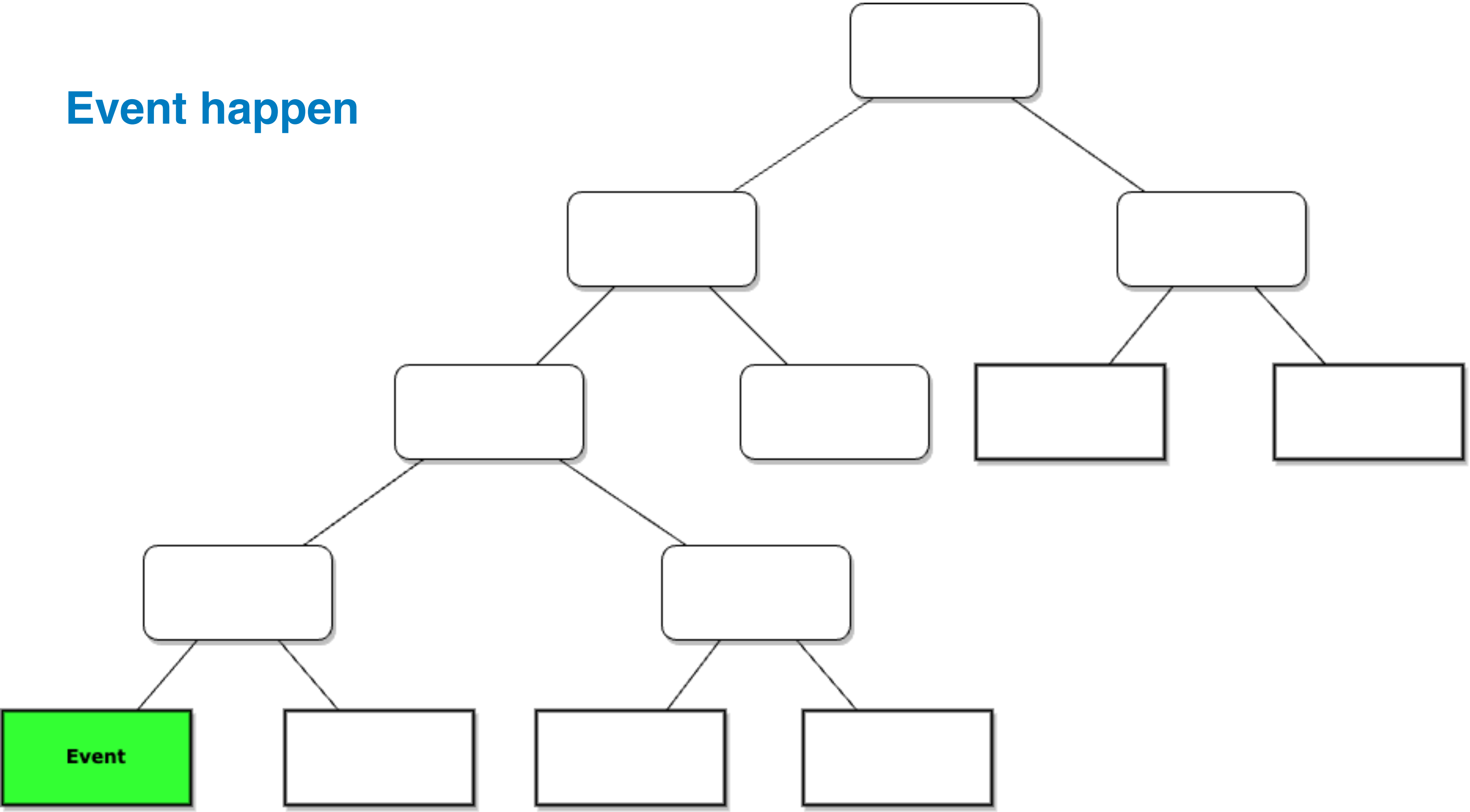
    *Can be influenced by OnPush

DOM Tree structure & data structure are tightly coupled

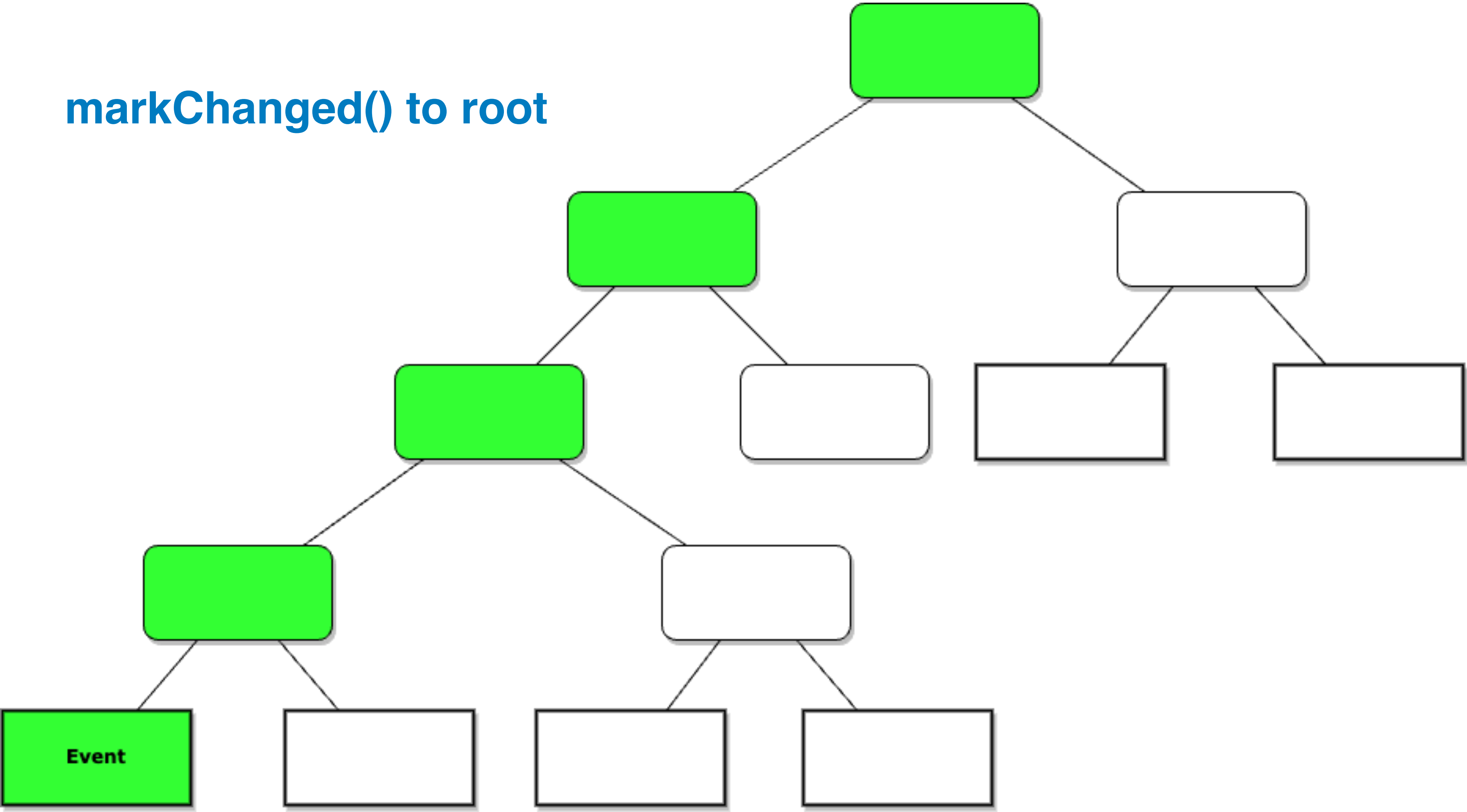
ZoneJS tree



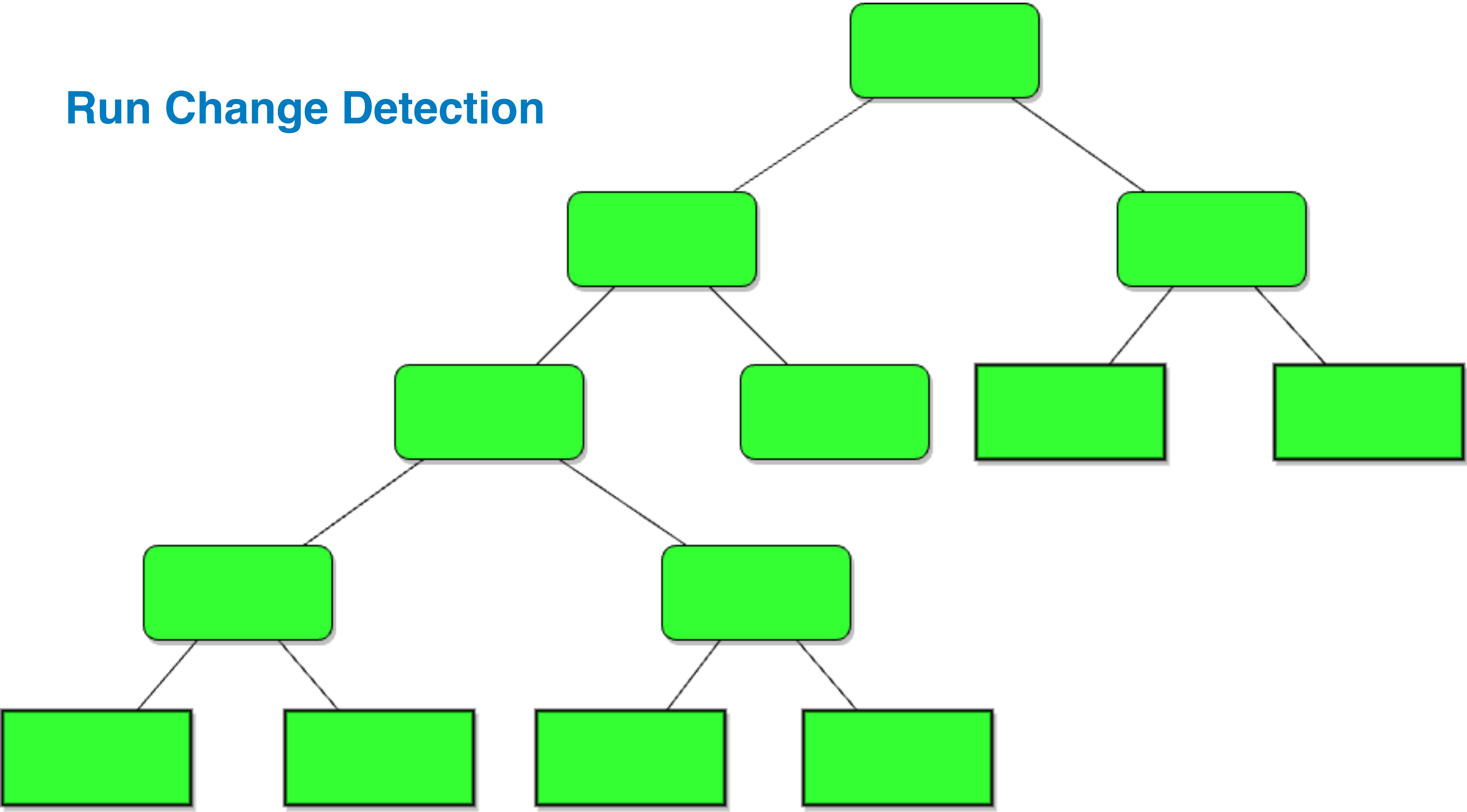
Event happen



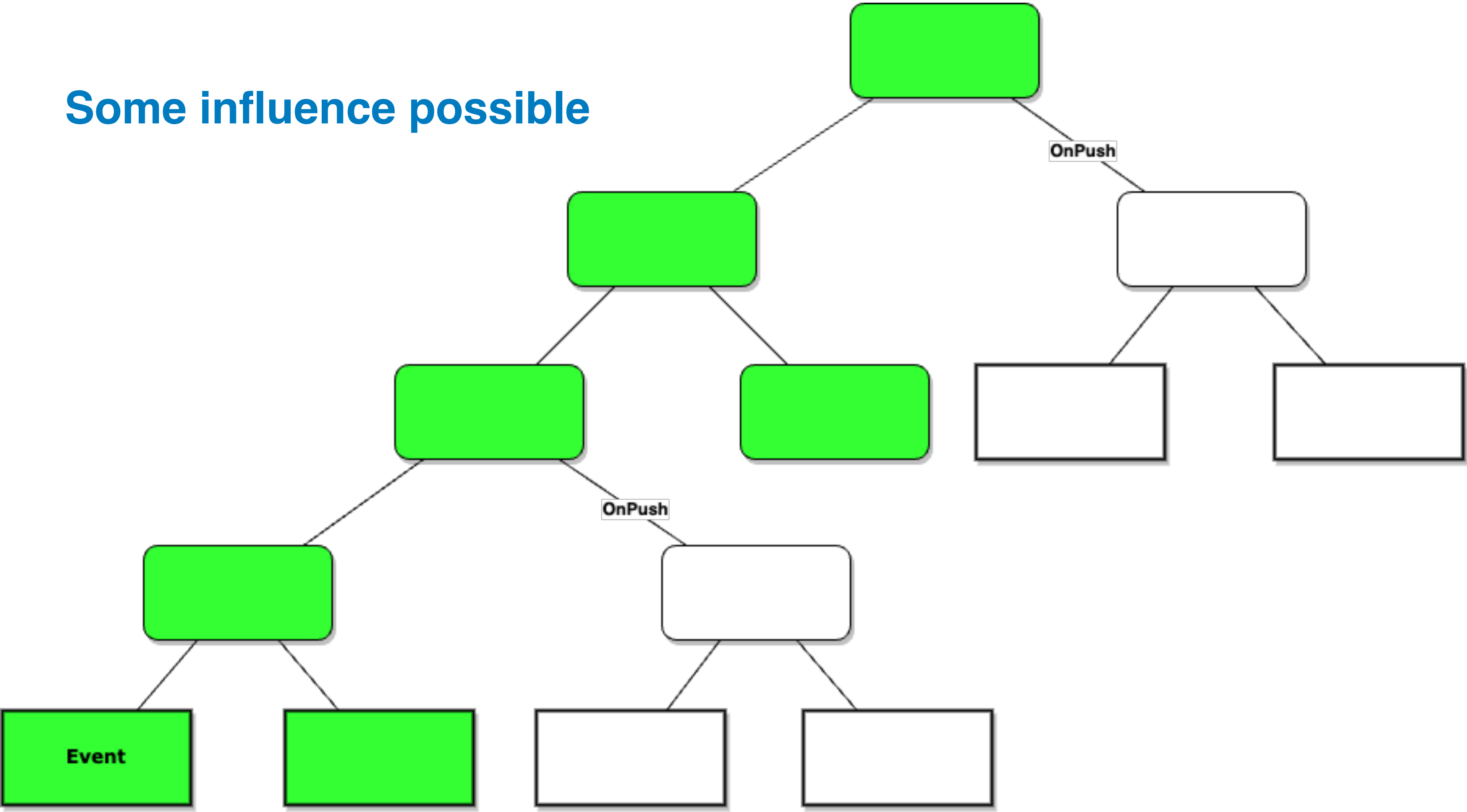
markChanged() to root



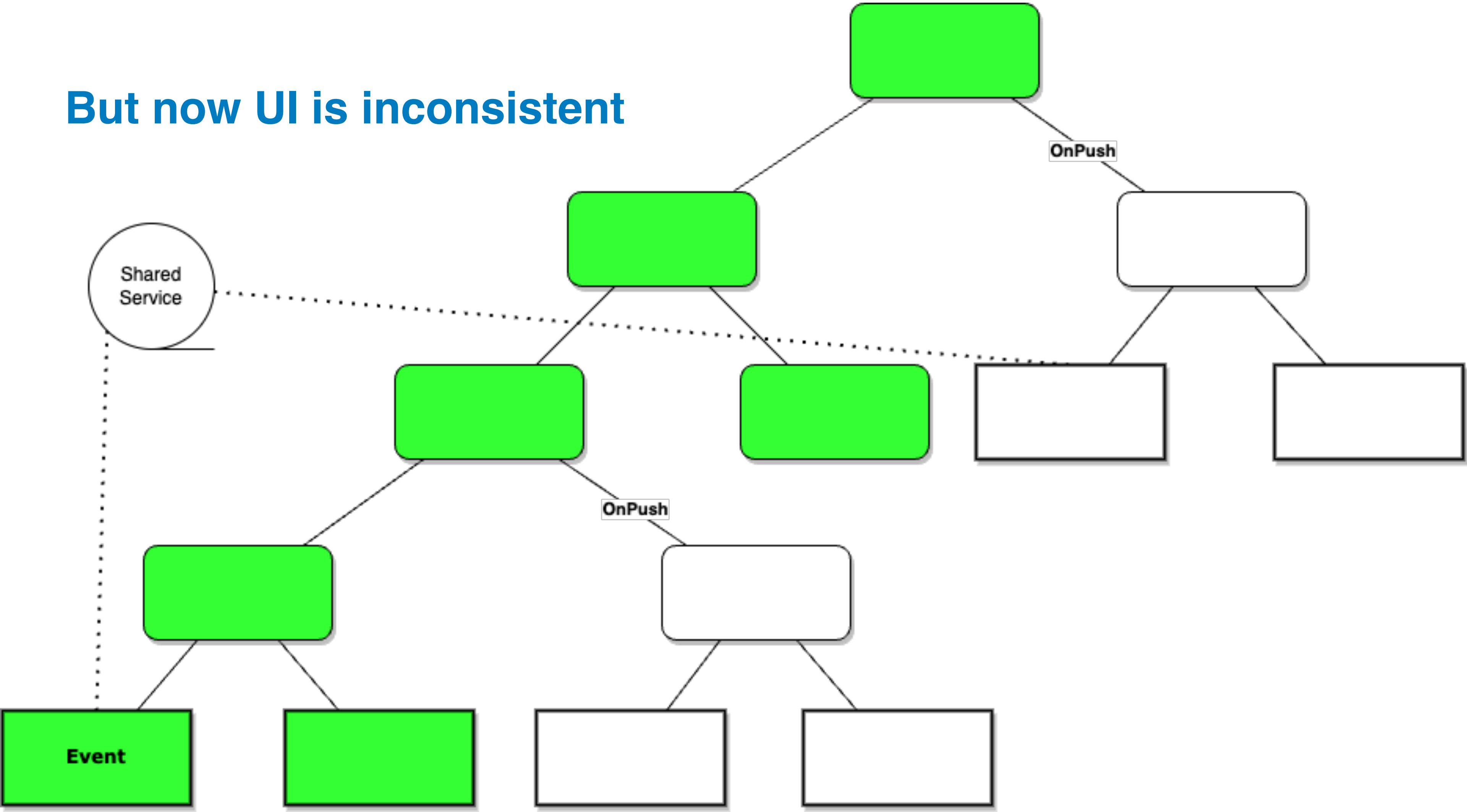
Run Change Detection



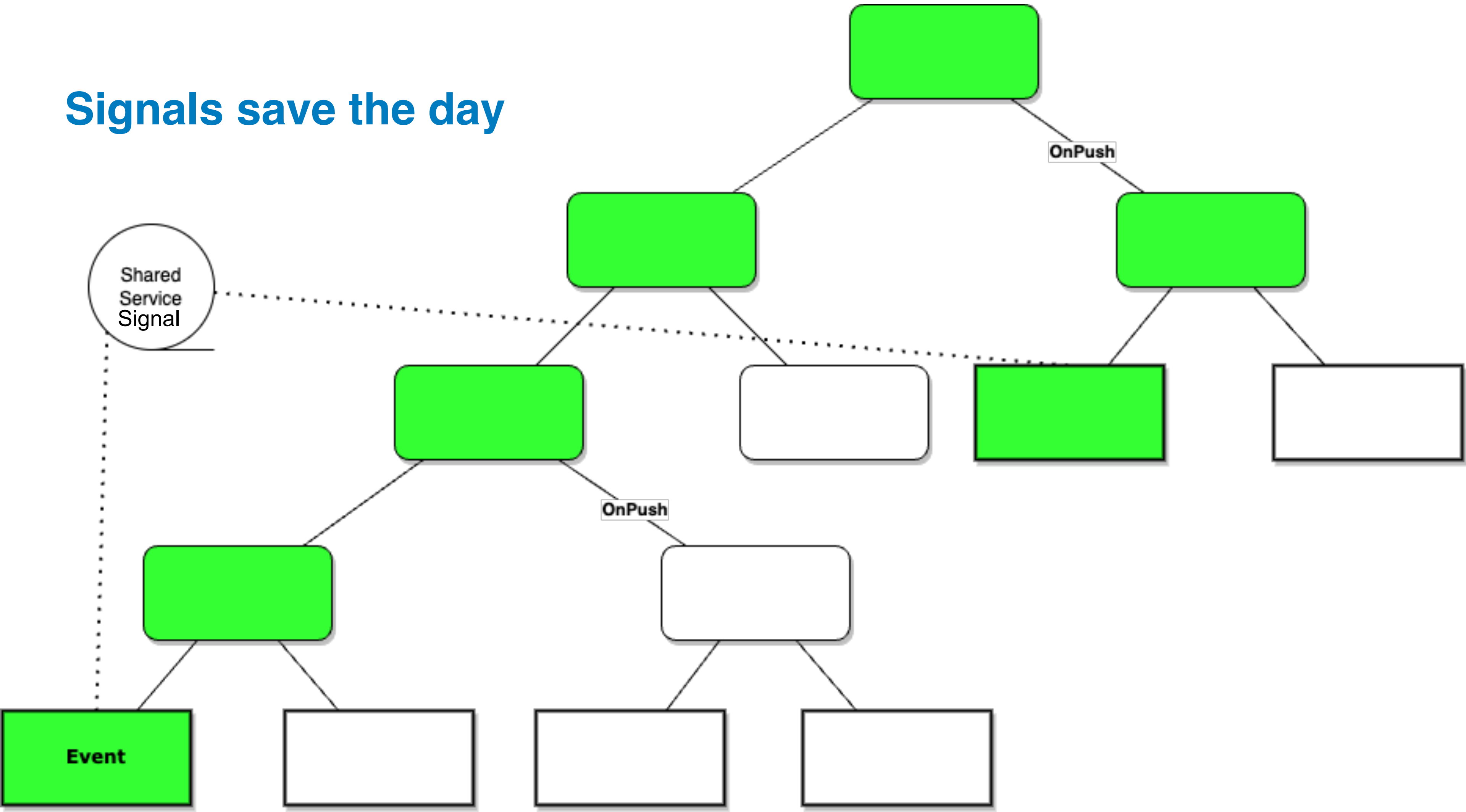
Some influence possible



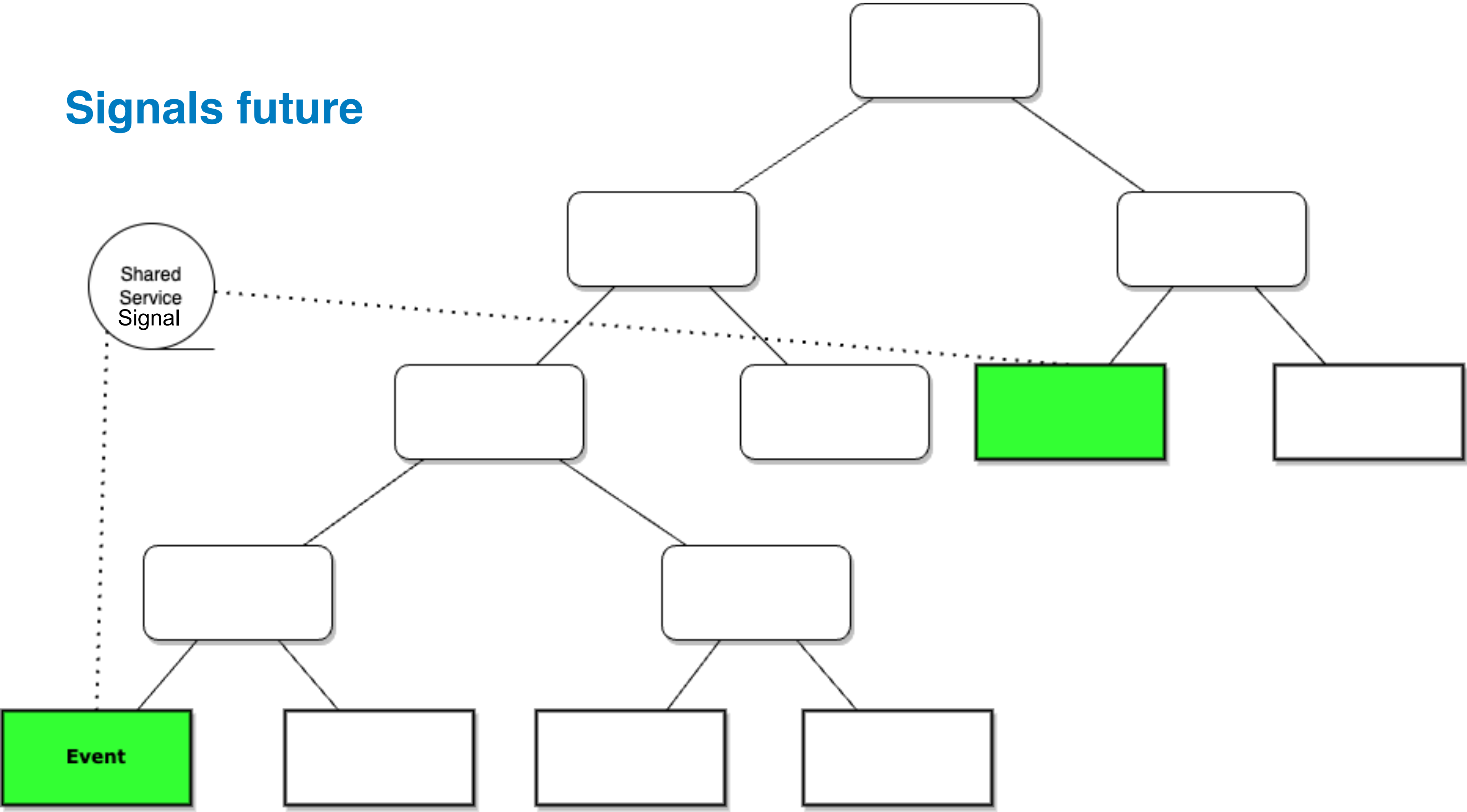
But now UI is inconsistent



Signals save the day



Signals future



Signals & RxJS

Will not replace RxJS

RxJS for async

Signals for everything else

Signals what?

RECONCEPT_

Signals: What

Reactive primitive

Synchronous

“Push” based

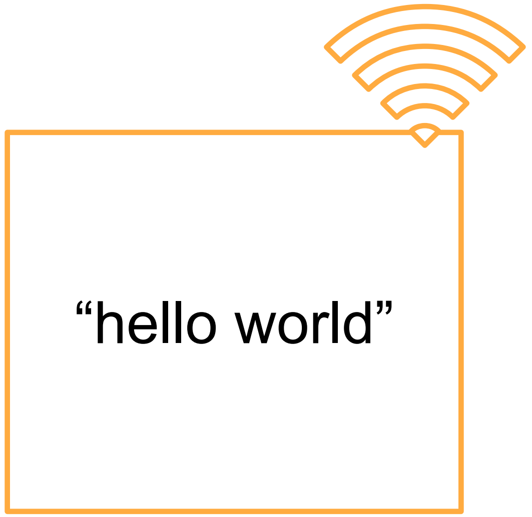
Signals: 3 primitives

```
signal(0)
```

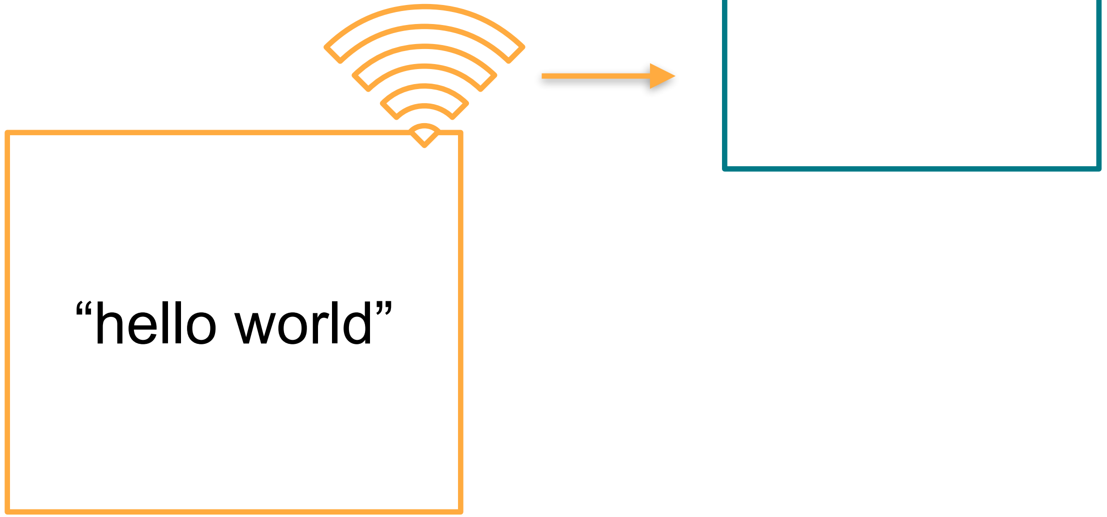
```
computed(
    () => {}
)
```

```
effect(
    () => {}
)
```

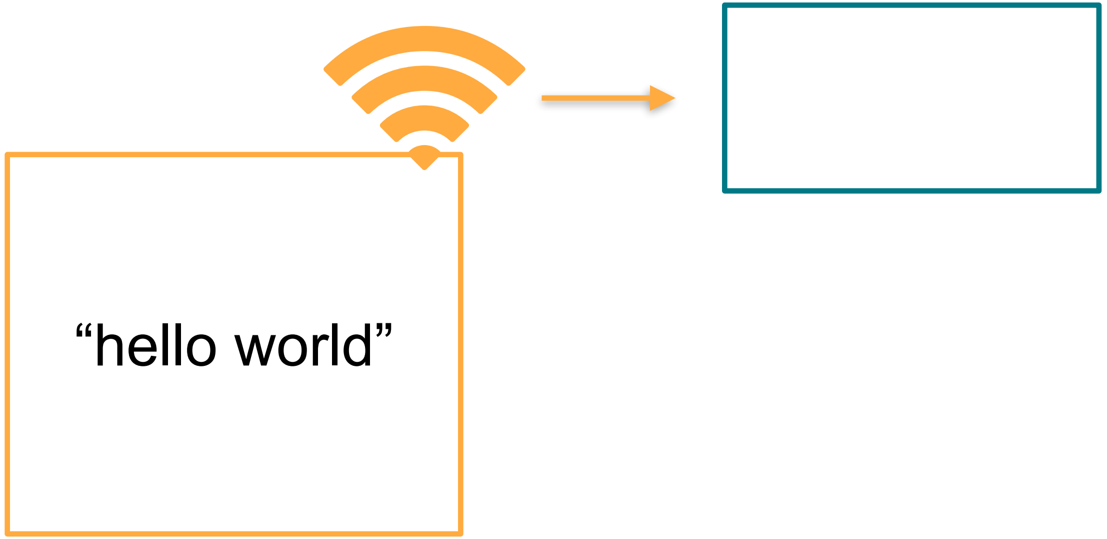
Signal: What - a box that holds a value



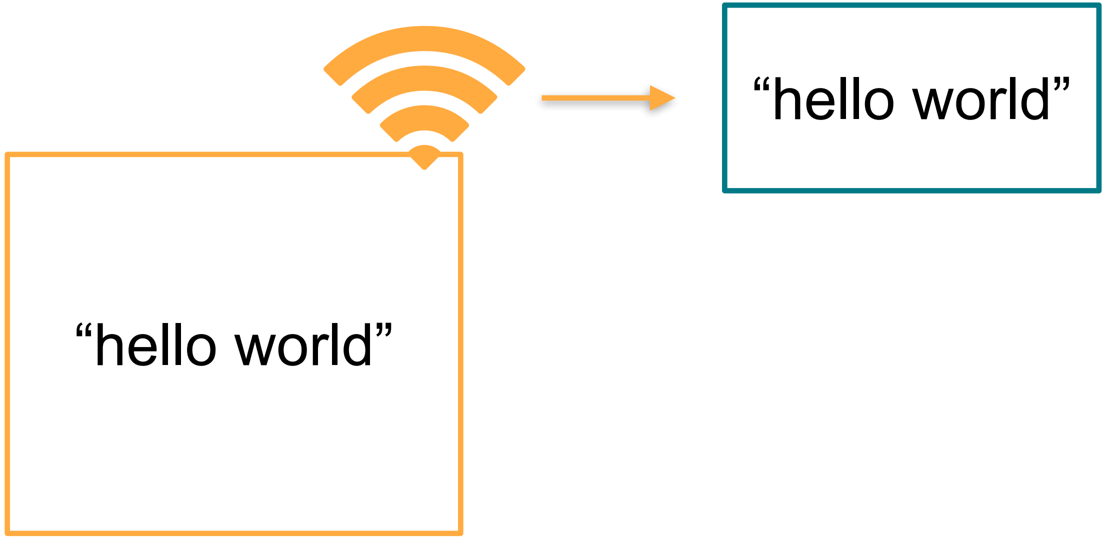
Signal: What - that can be listened to



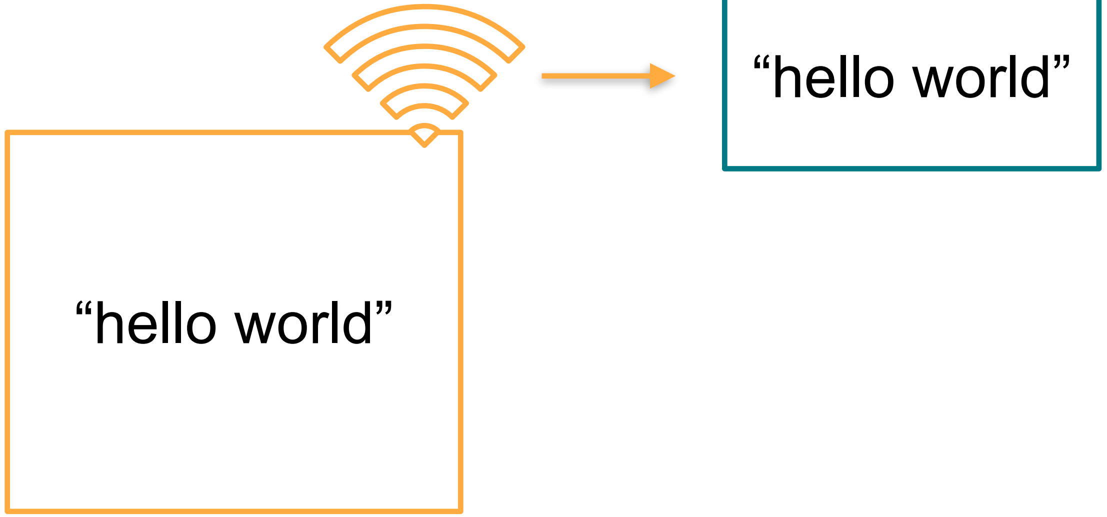
Signal: What - that can be listened to



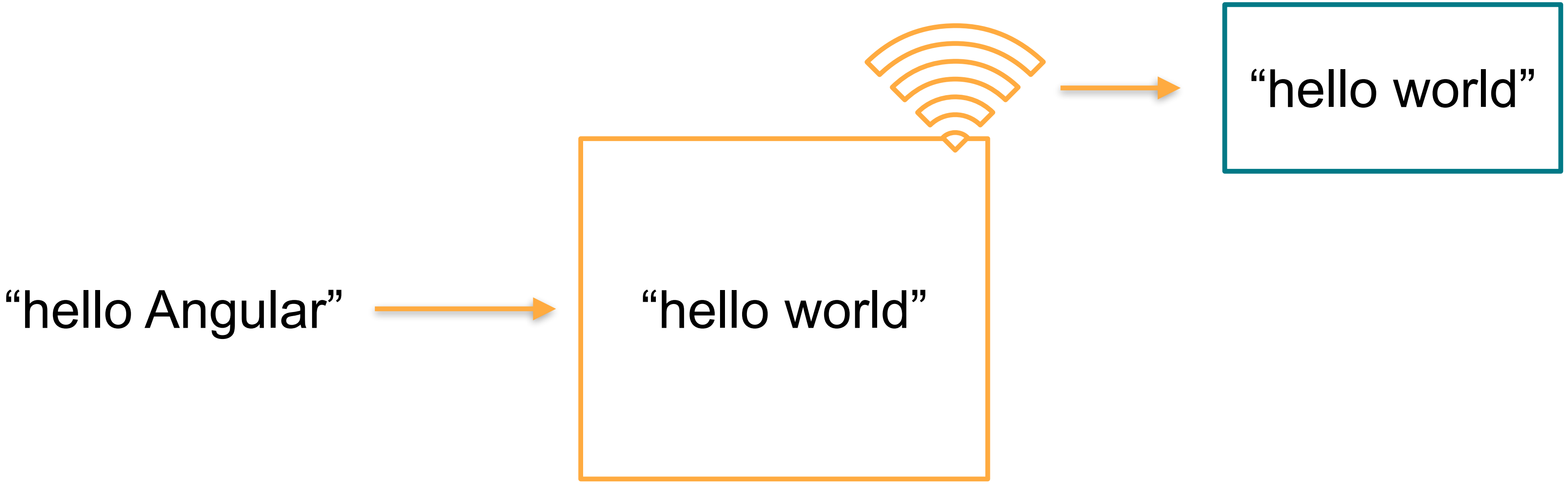
Signal: What - that can be listened to



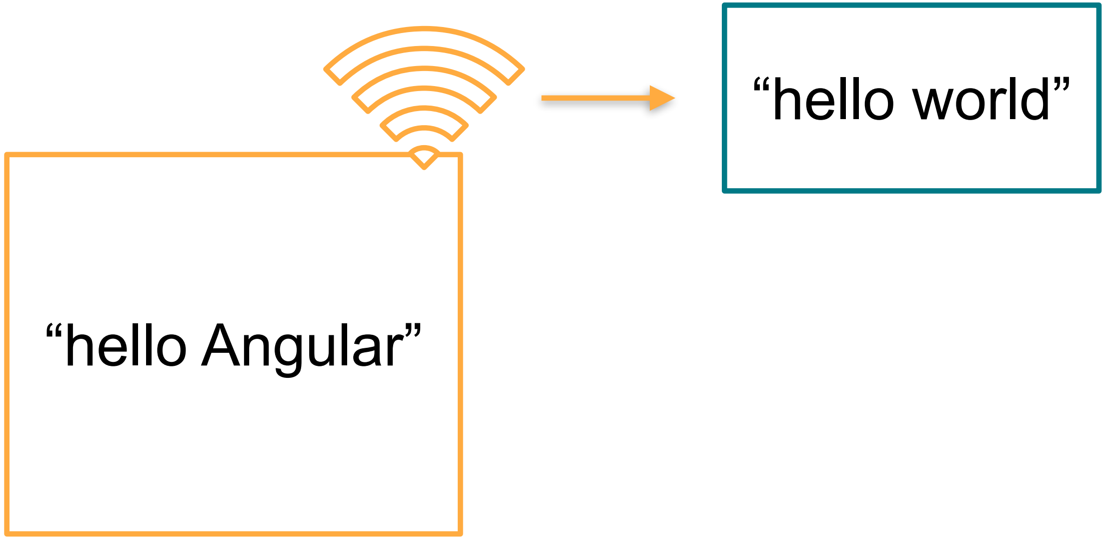
Signal: What - that can be listened to



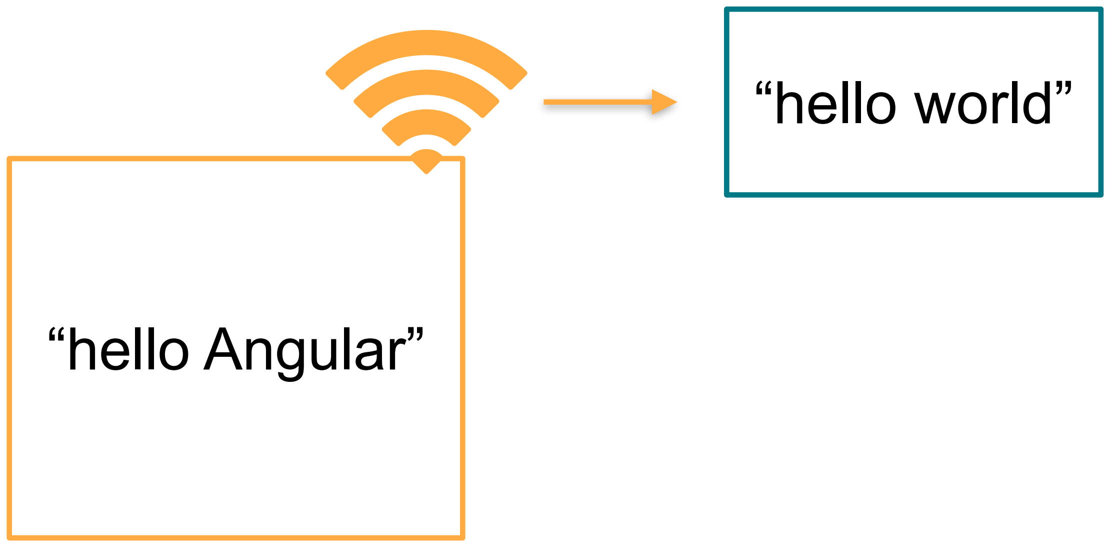
Signal: What - that can be changed*



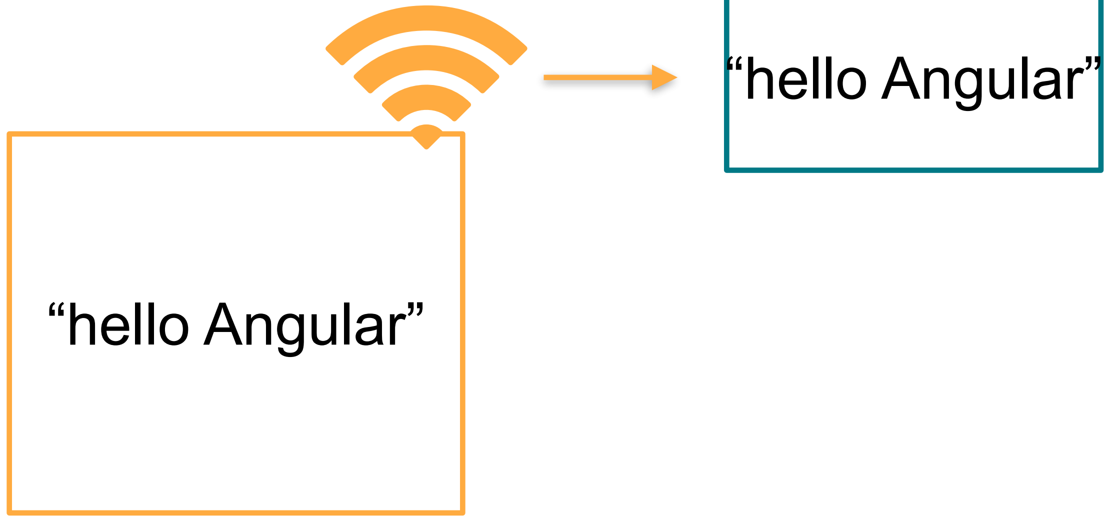
Signal: What - that can be changed*



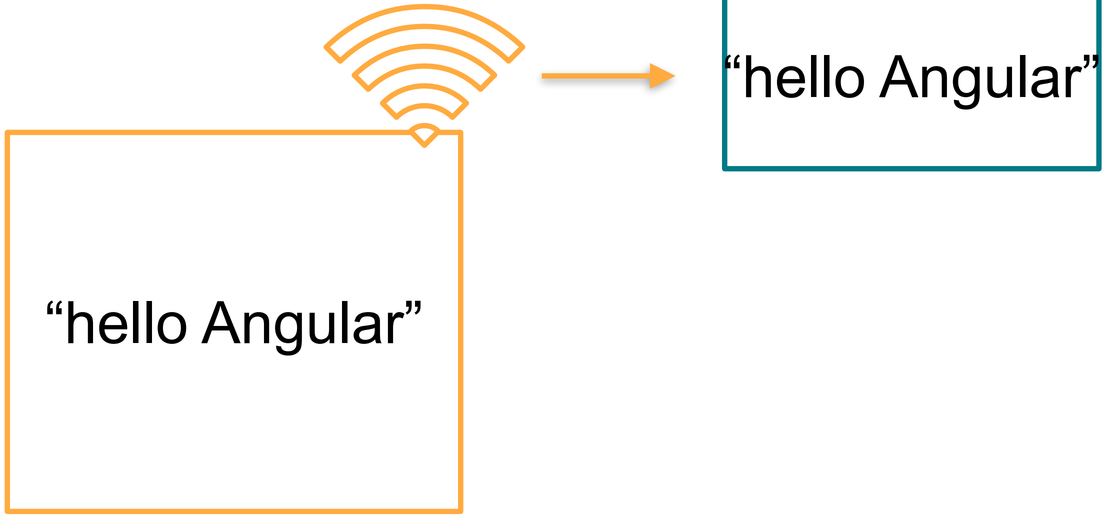
Signal: What - that can be changed*



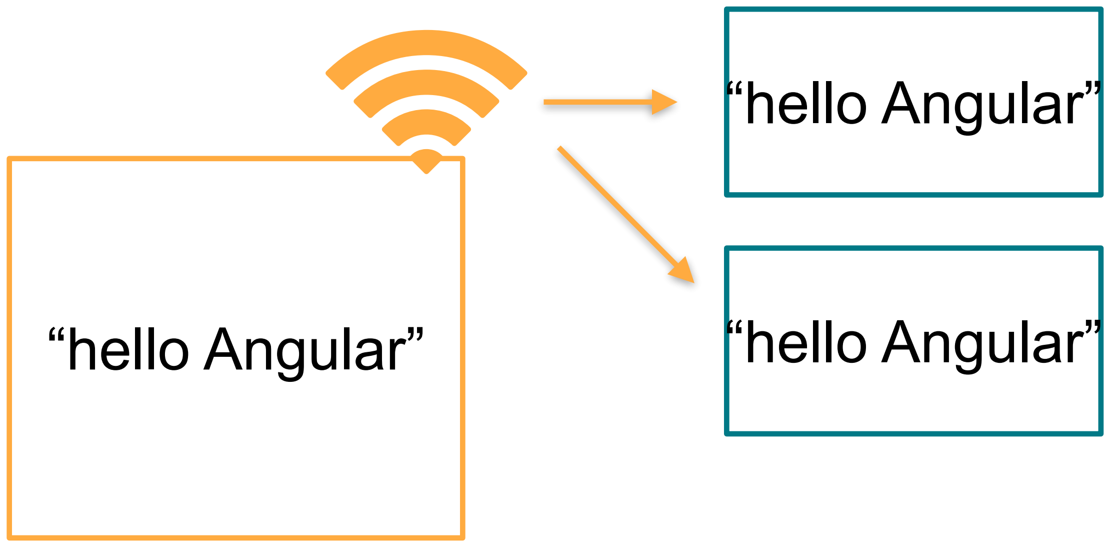
Signal: What - that can be changed*



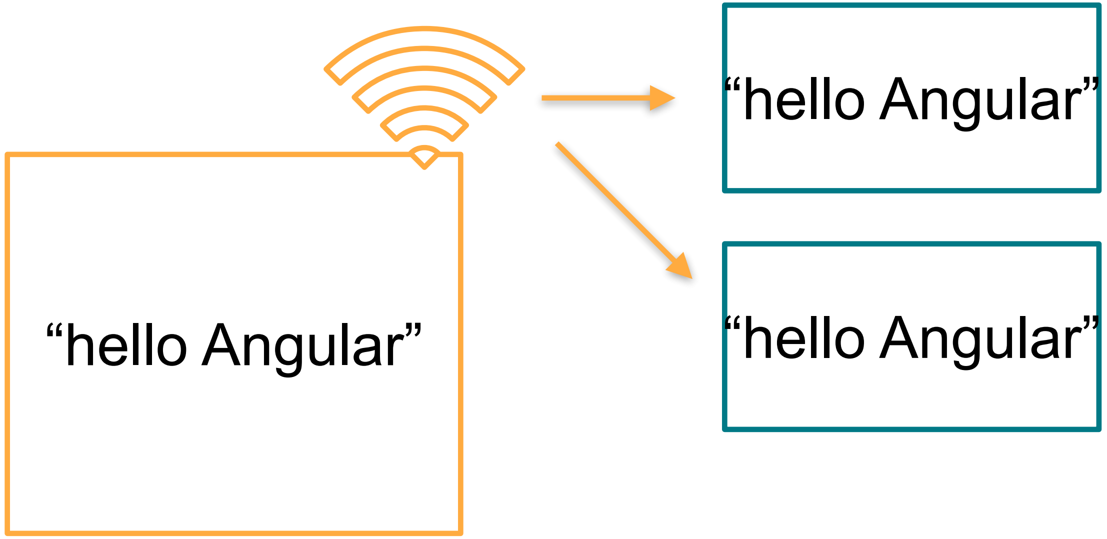
Signal: What - that can be changed*



Signal: What - have multiple listeners



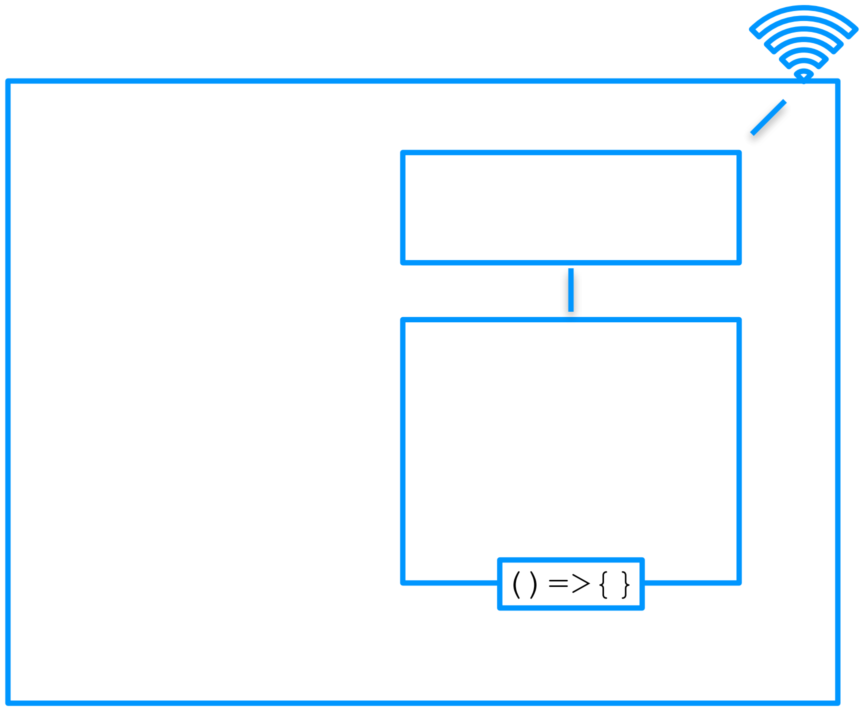
Signal: What - have multiple listeners



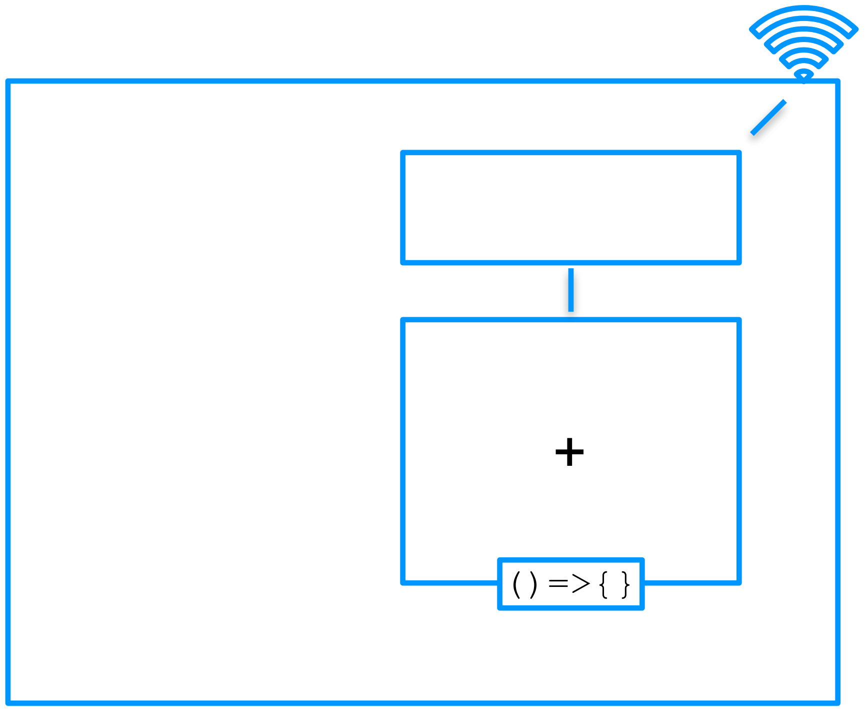
Signals Demo

RECONCEPT_

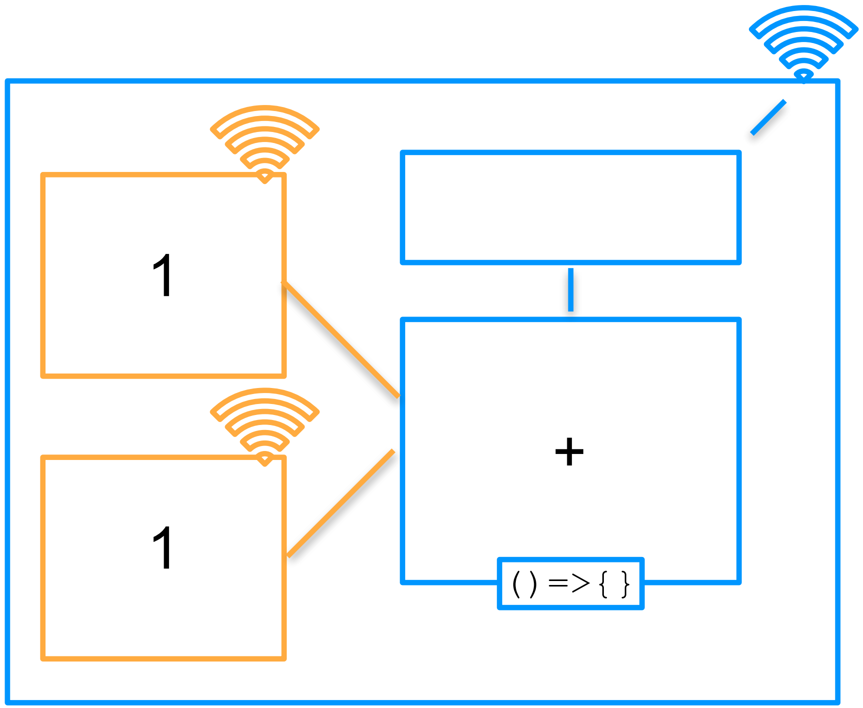
Computed: What - a signal that holds an operation



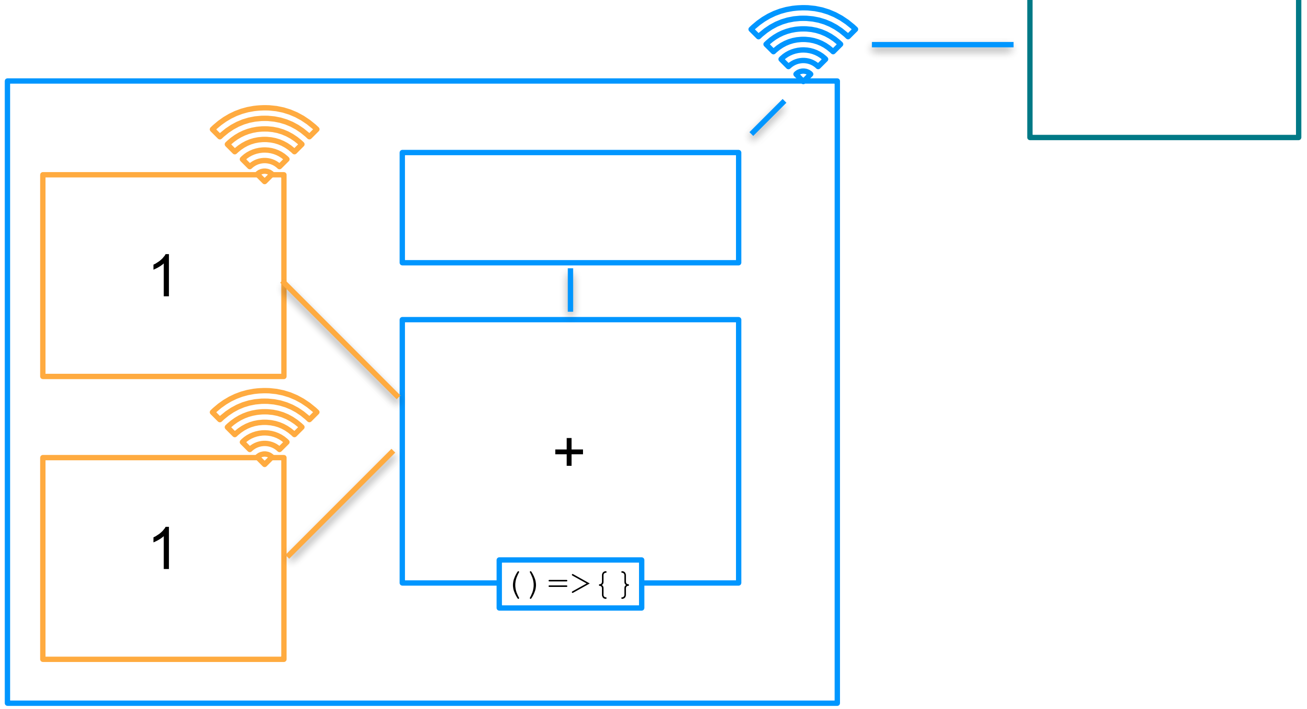
Computed: What - a signal that holds an operation



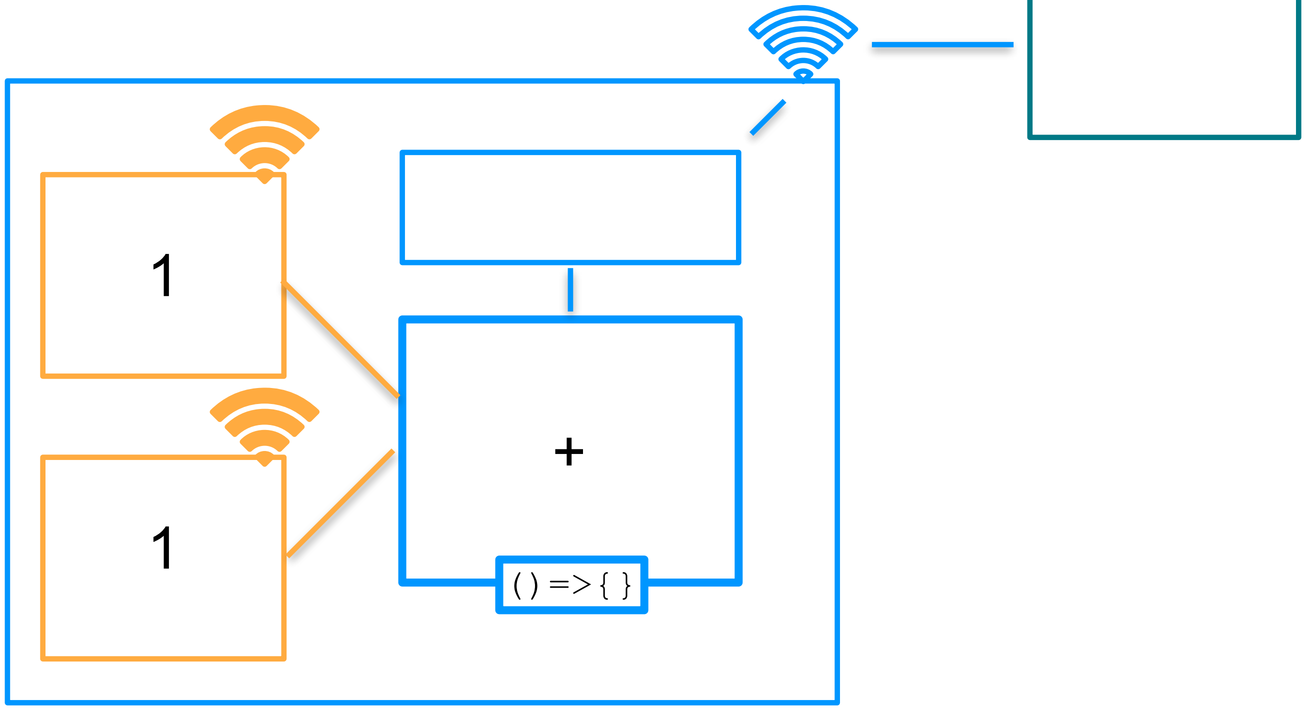
Computed: What - listens to other signals



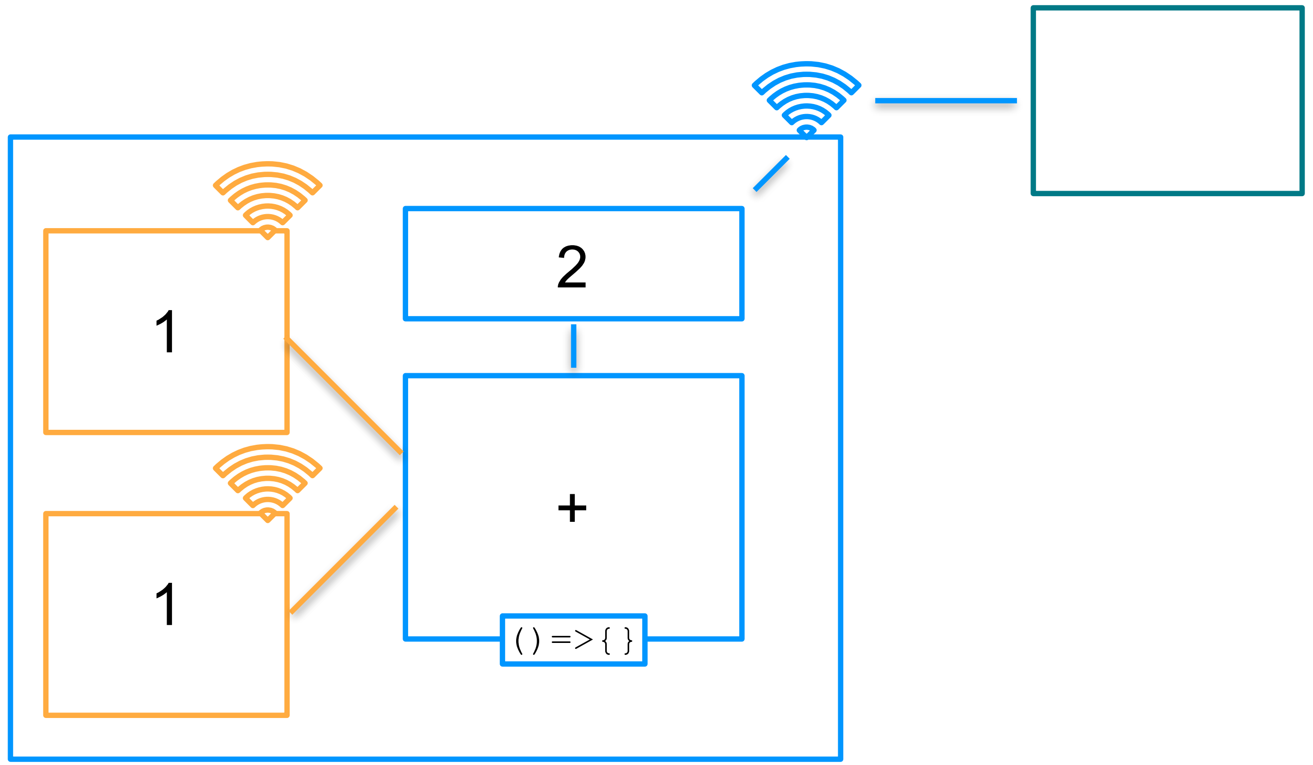
Computed: What - Executed when listened to



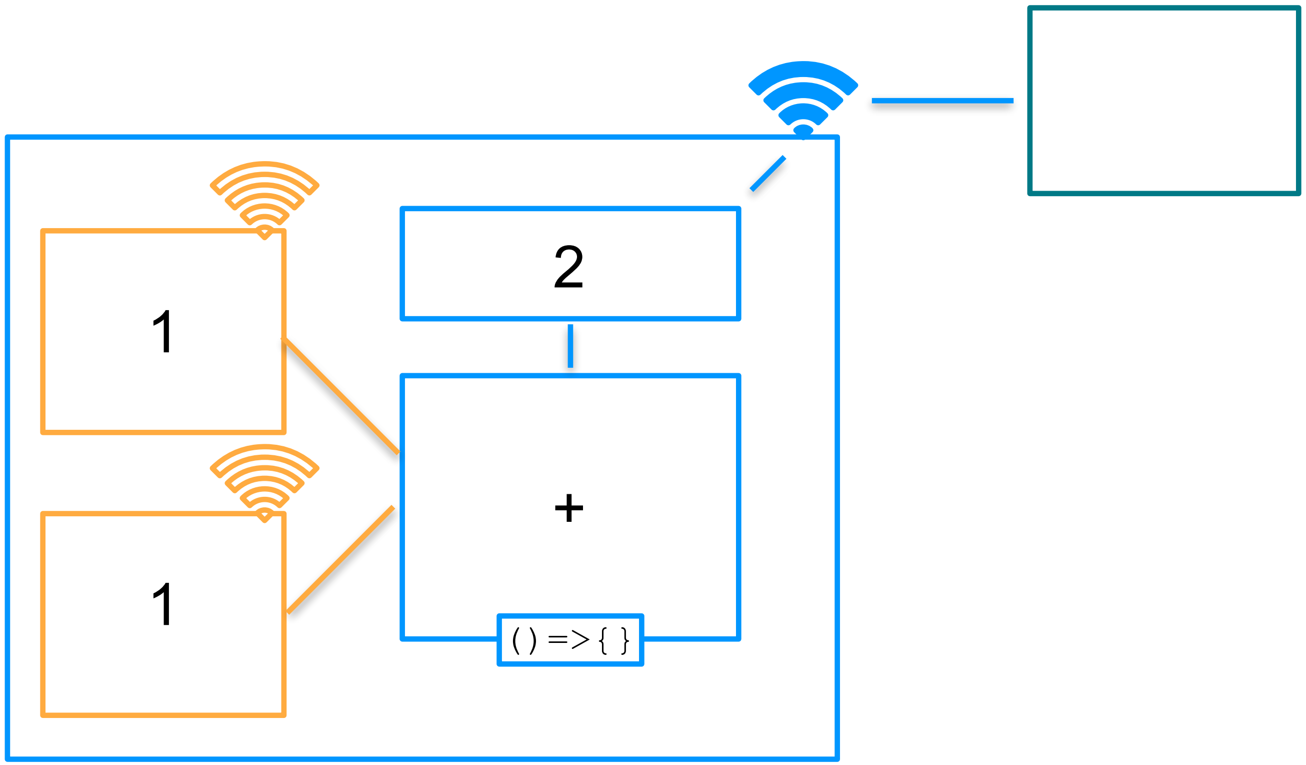
Computed: What - Executed when listened to



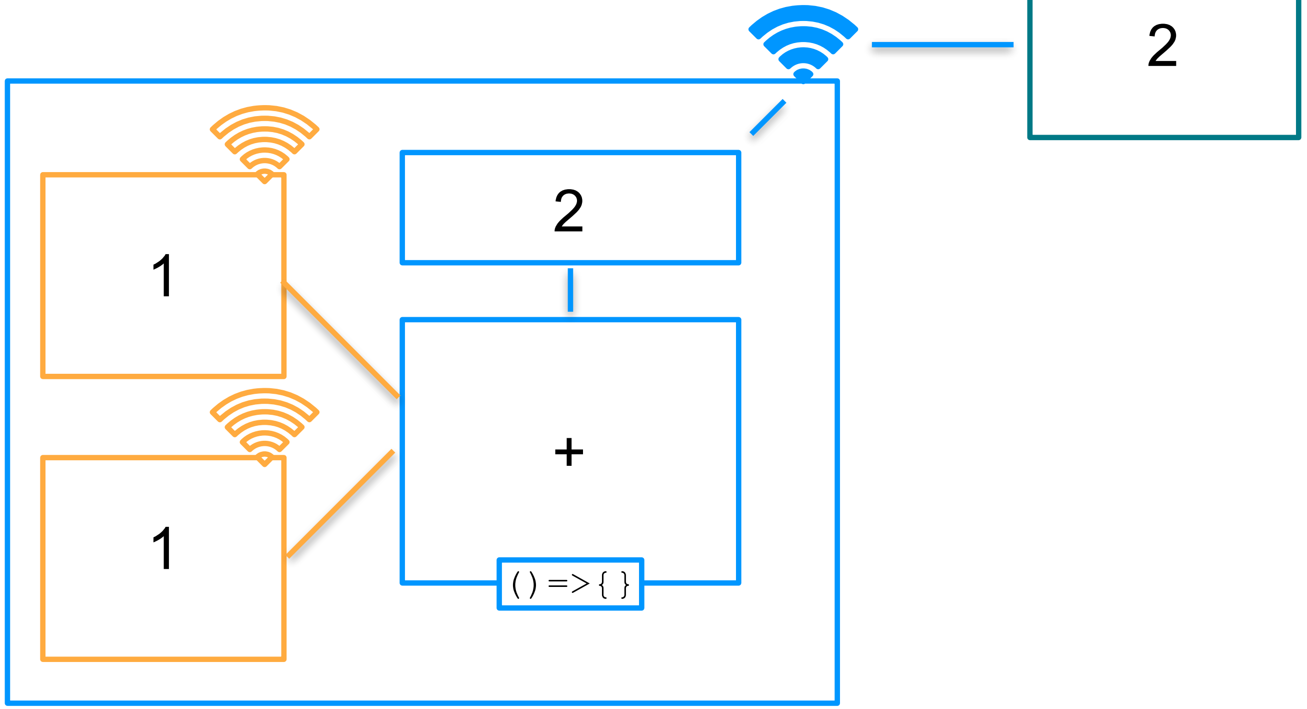
Computed: What - Executed when listened to



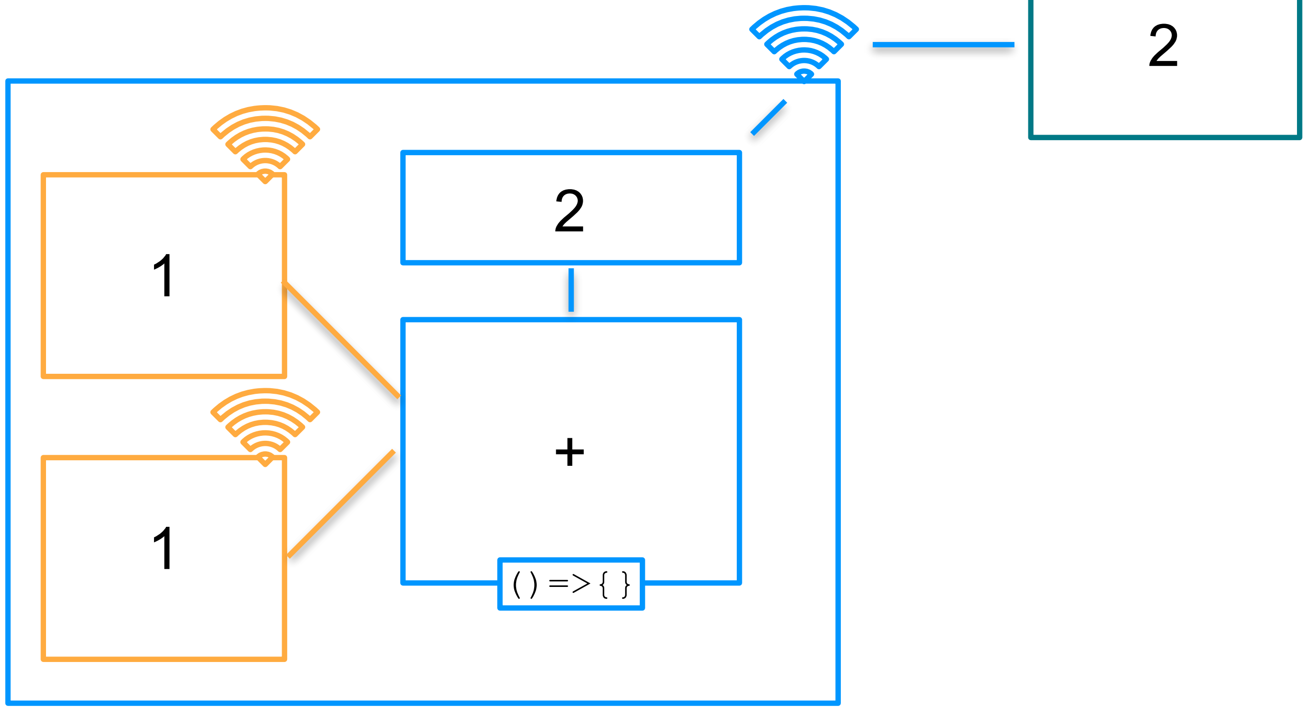
Computed: What - Executed when listened to



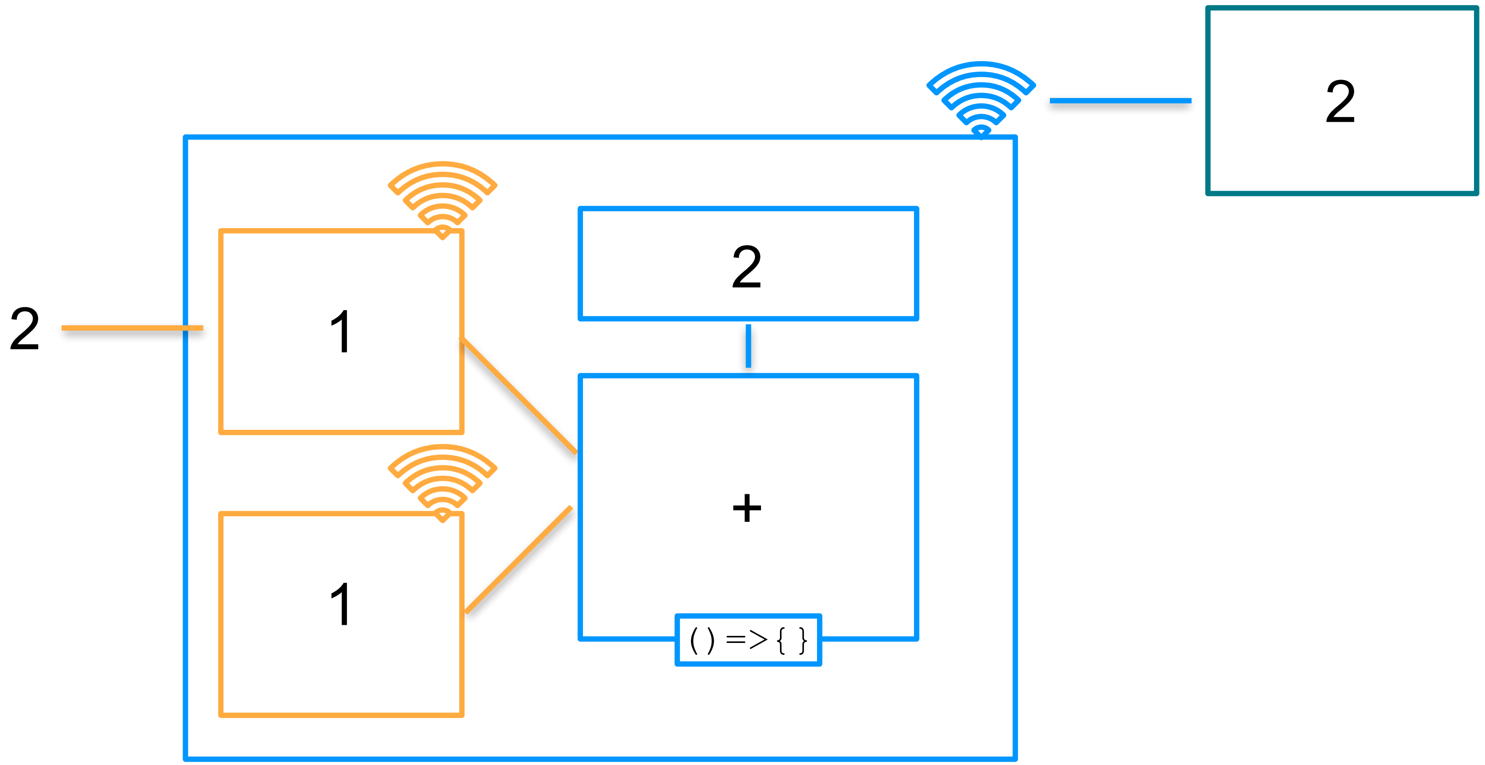
Computed: What - Executed when listened to



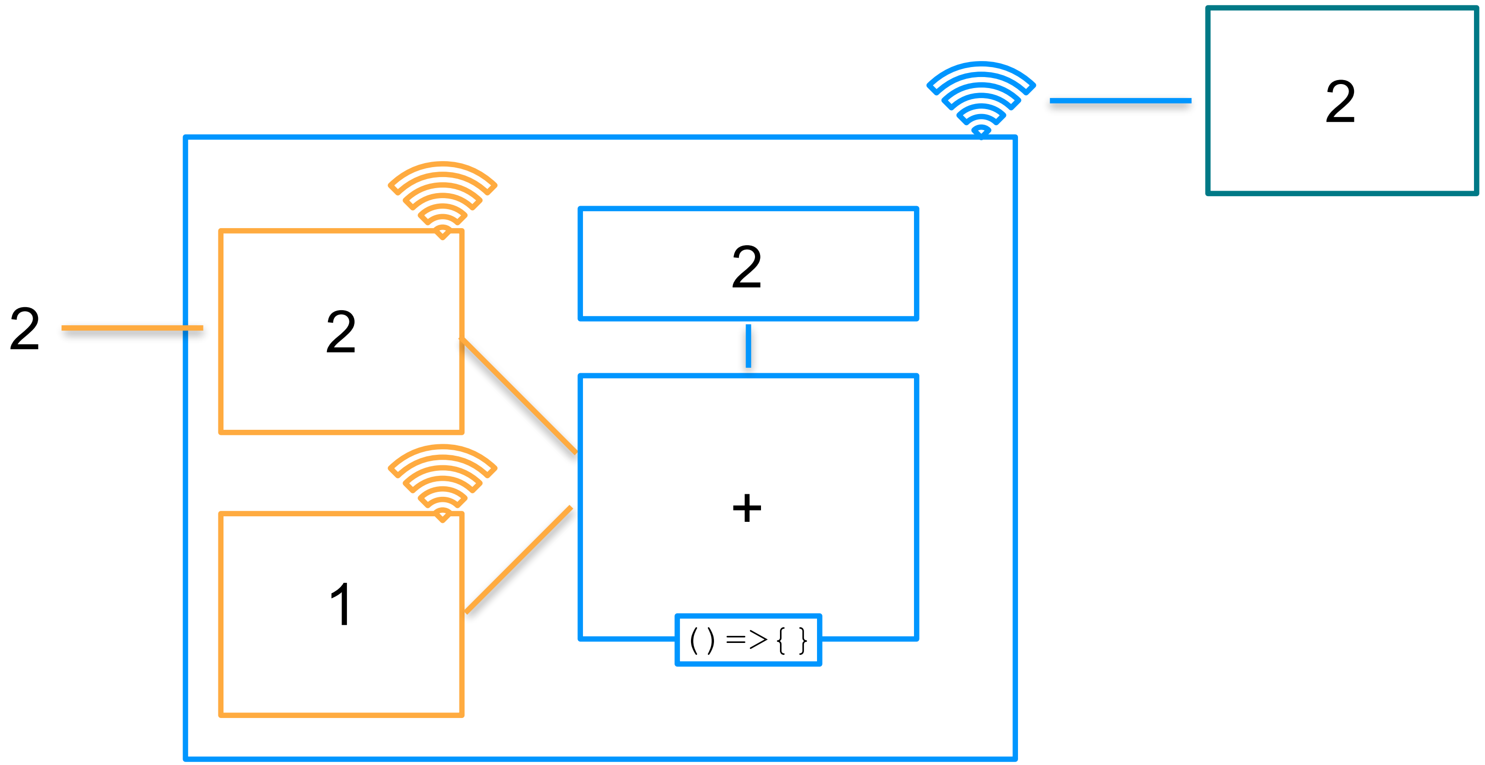
Computed: What - Executed when listened to



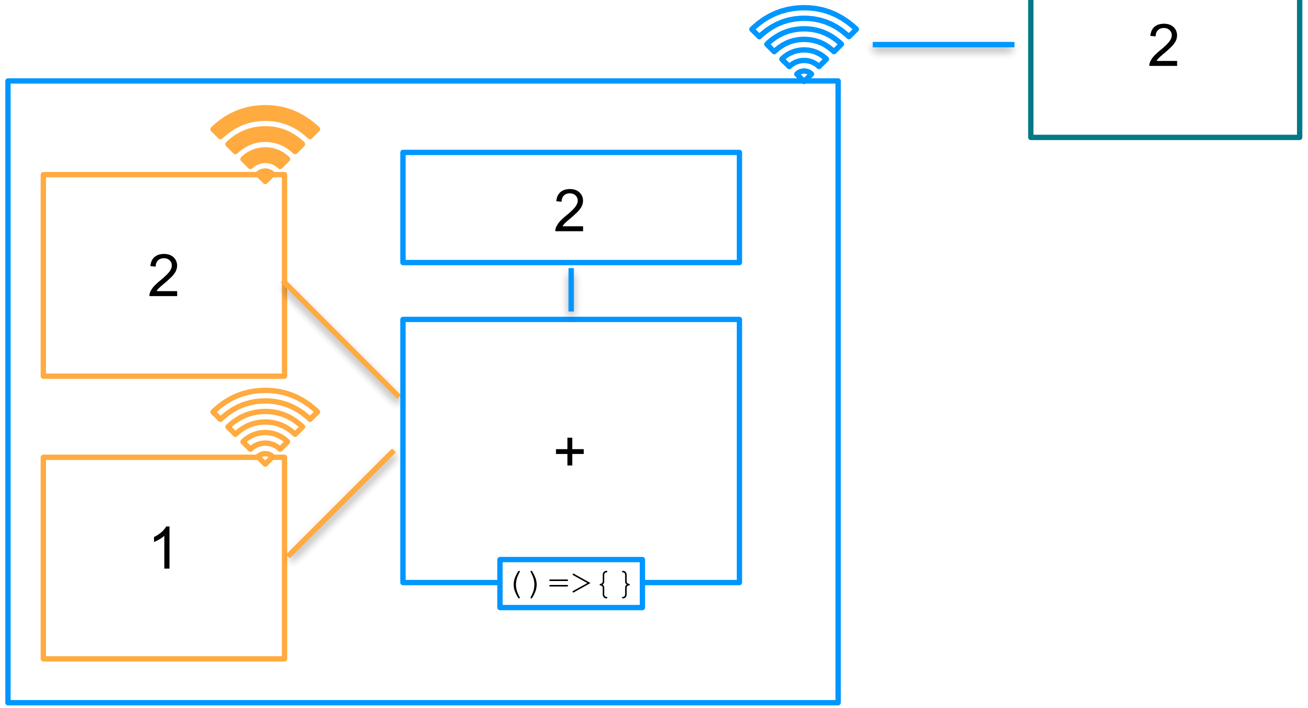
Computed: What - when *any* signal changes



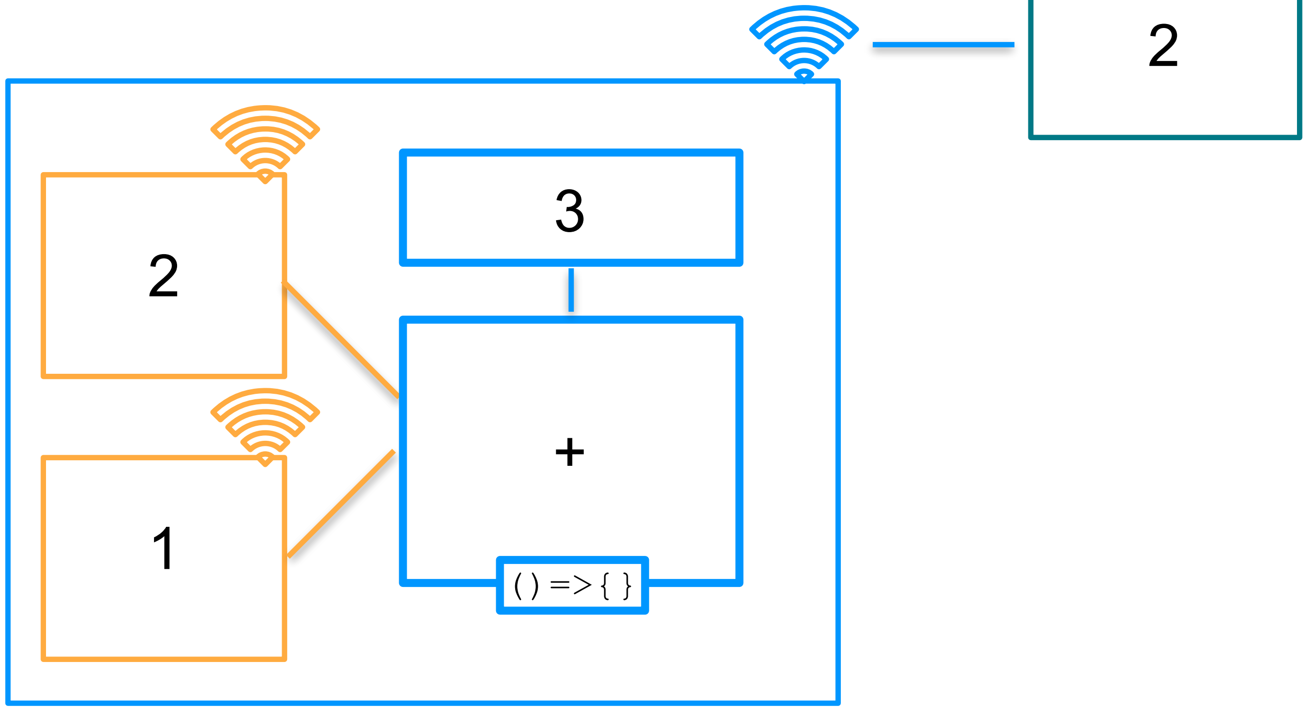
Computed: What - when *any* signal changes



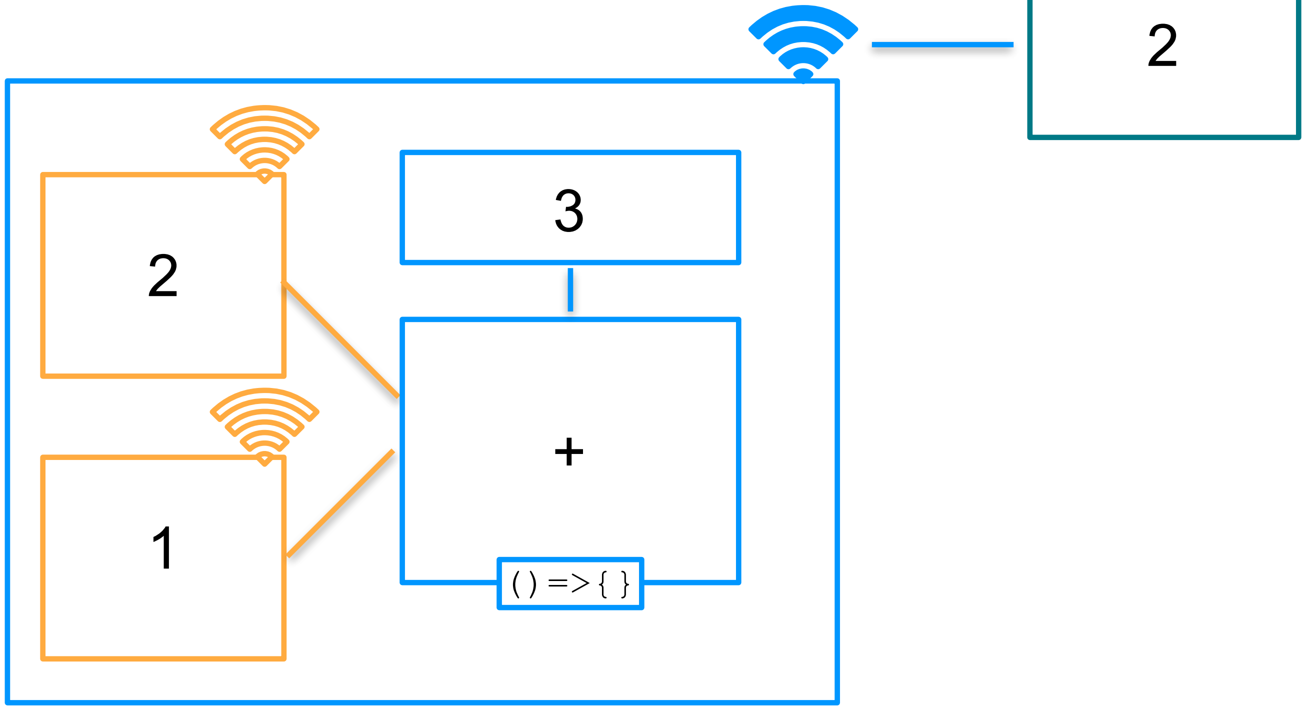
Computed: What - when *any* signal changes



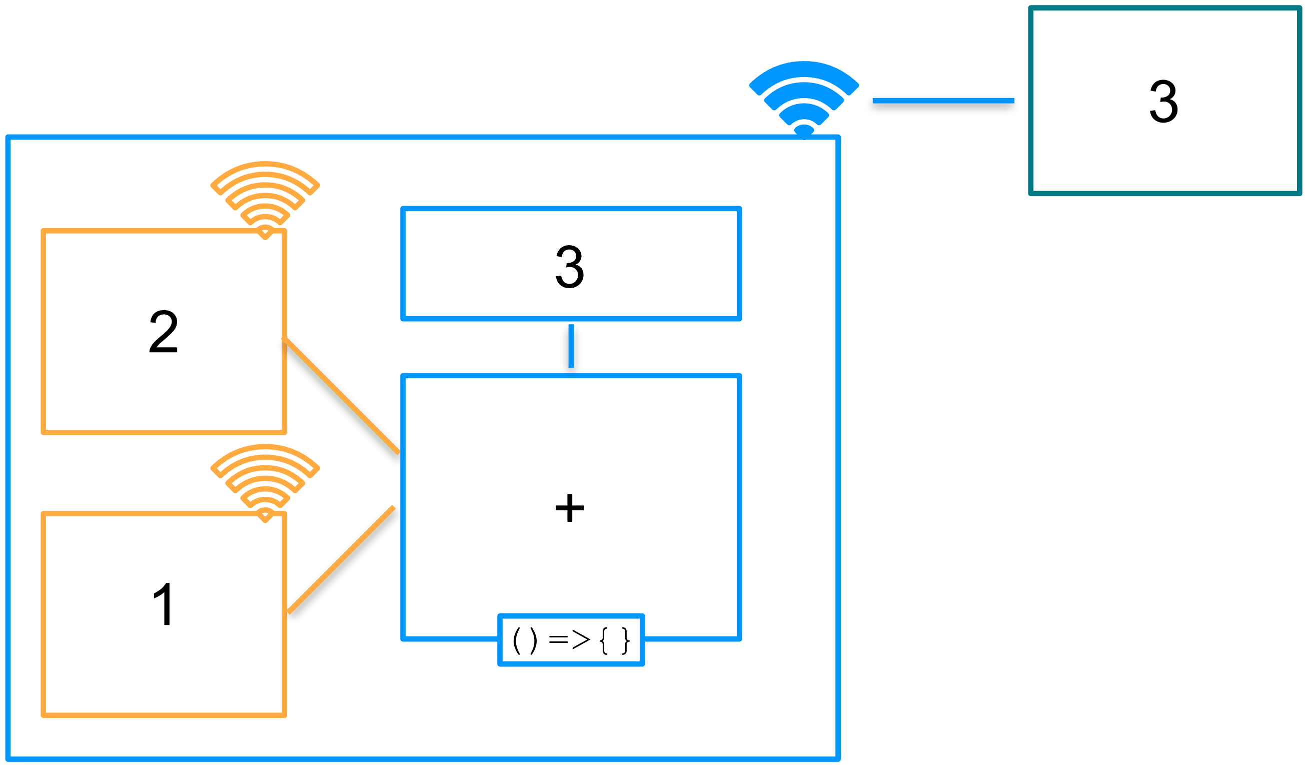
Computed: What - The operation is run again



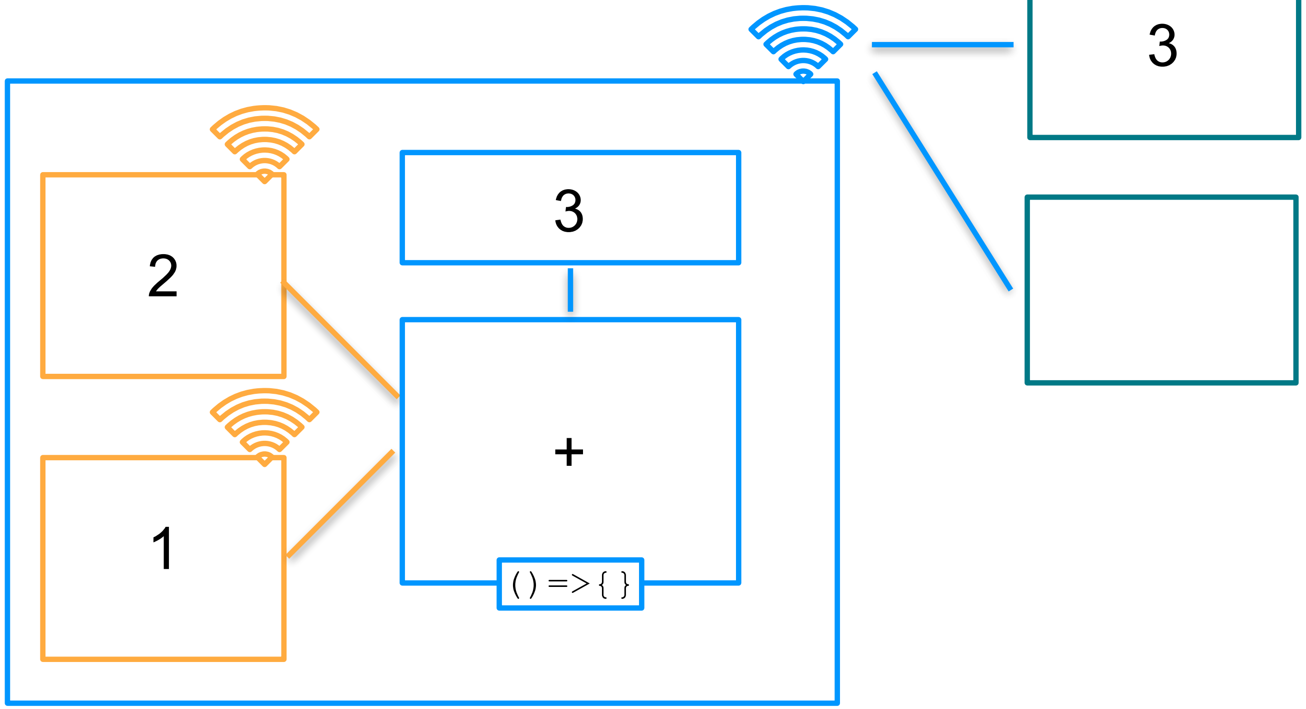
Computed: What - And listeners are notified



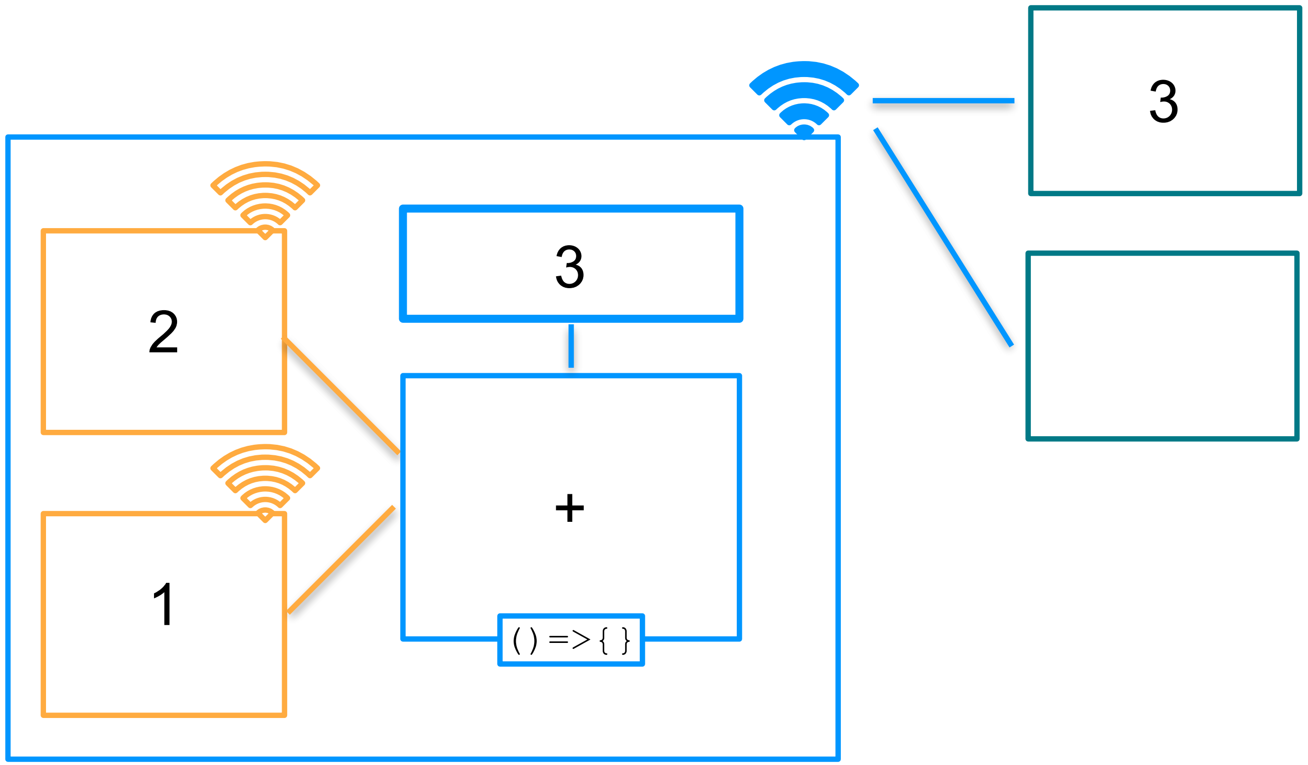
Computed: What - And listeners are notified



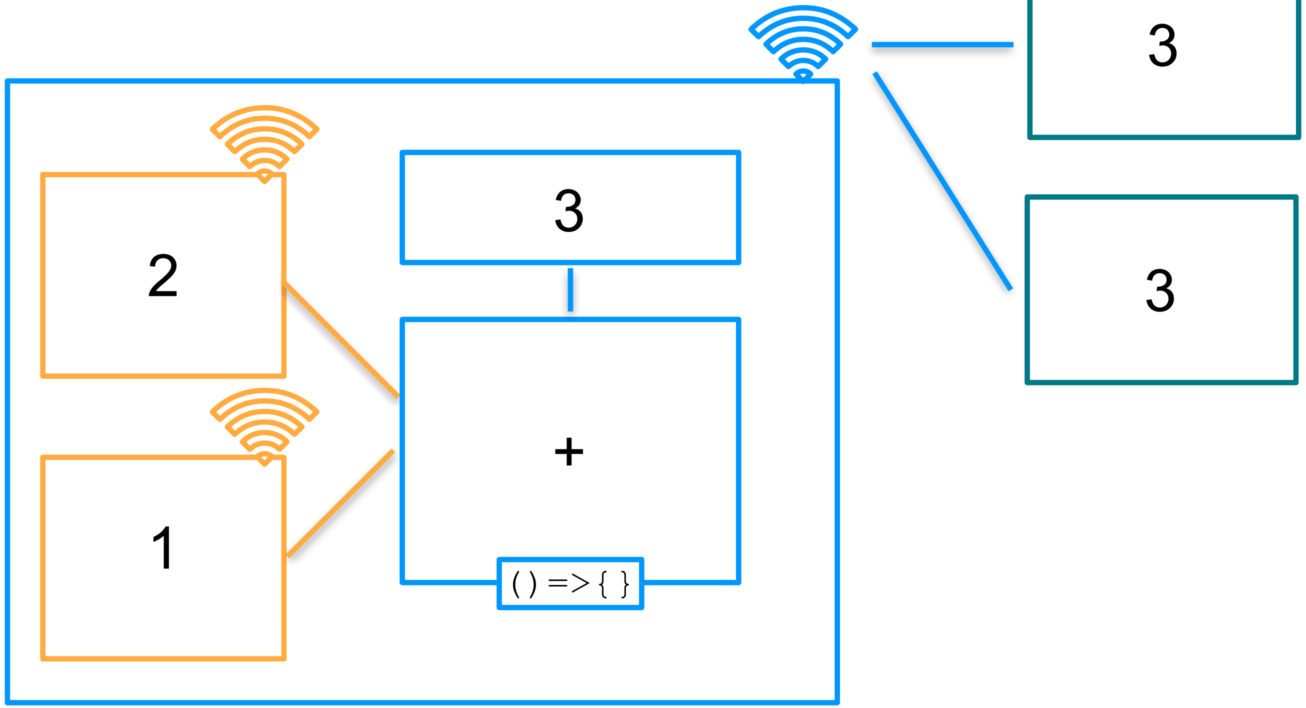
Computed: What - More listeners will get the latest value



Computed: What - More listeners will get the latest value



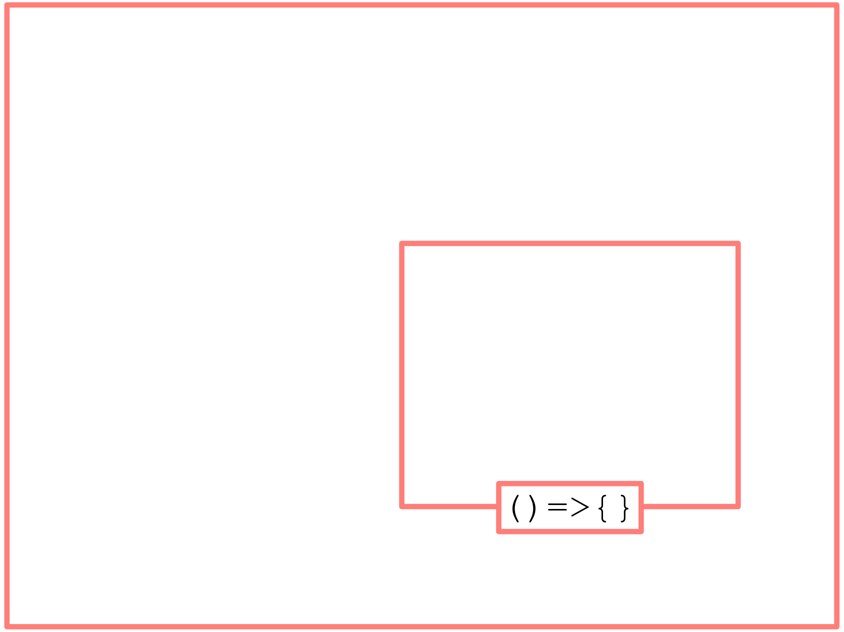
Computed: What - More listeners will get the latest value



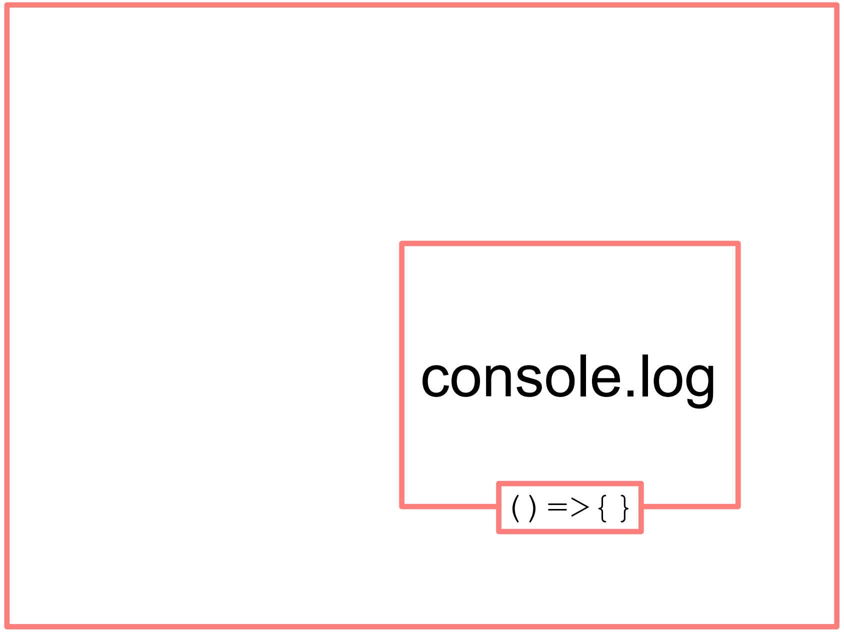
Computed Demo

RECONCEPT_

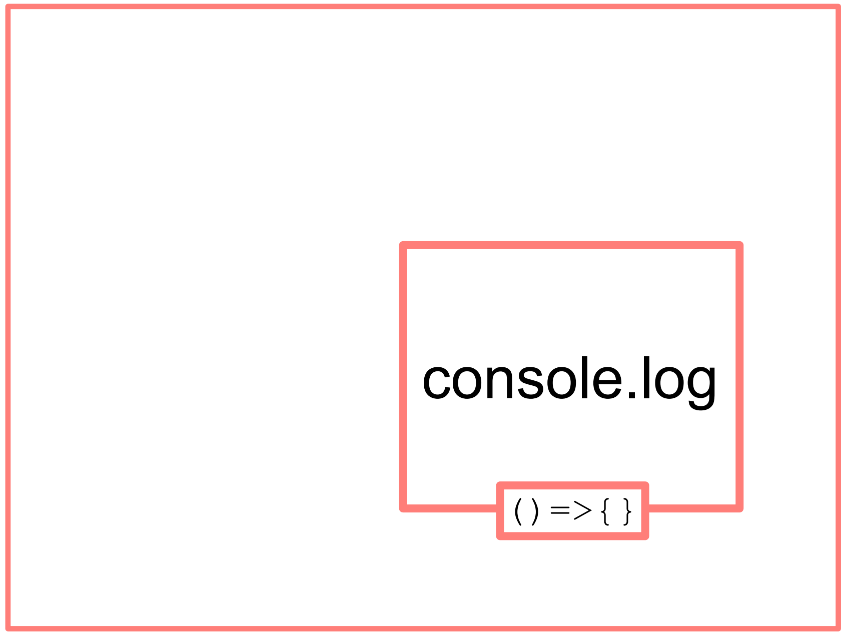
Effect: What - side effects



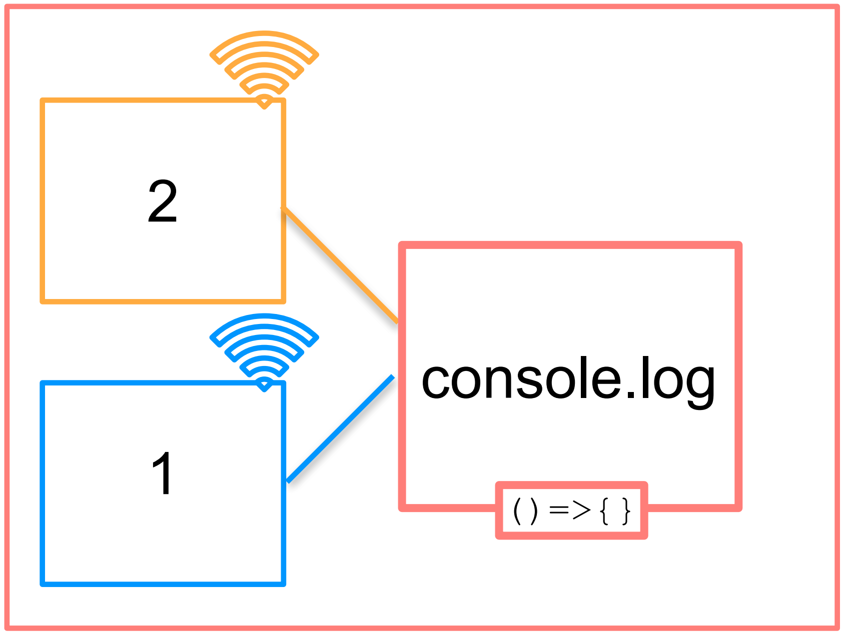
Effect: What - side effects



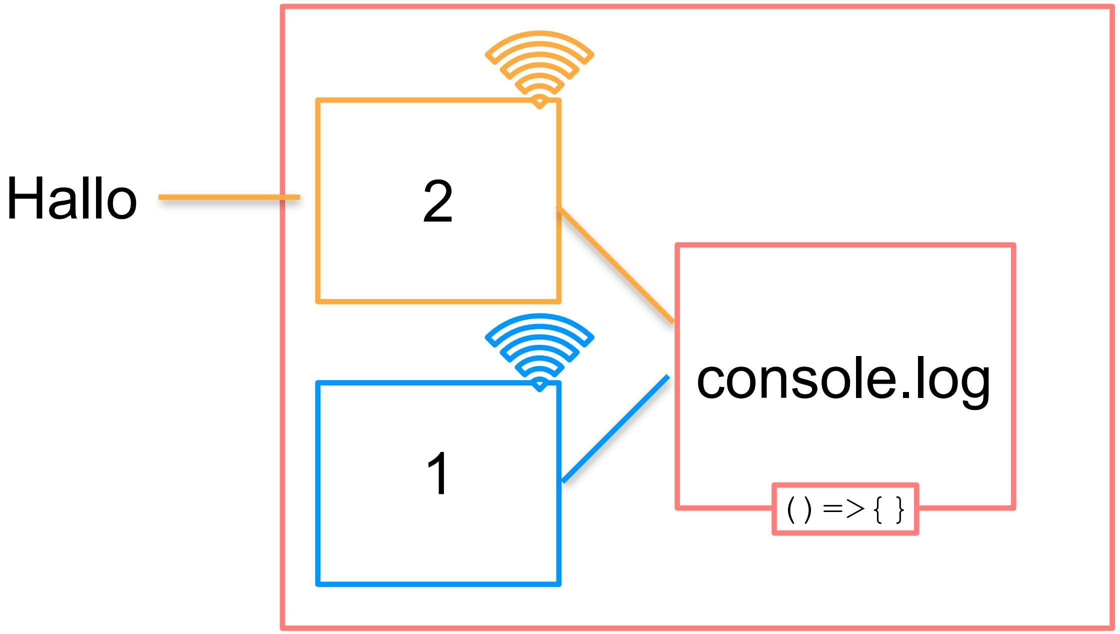
Effect: What - side effects, run immediately



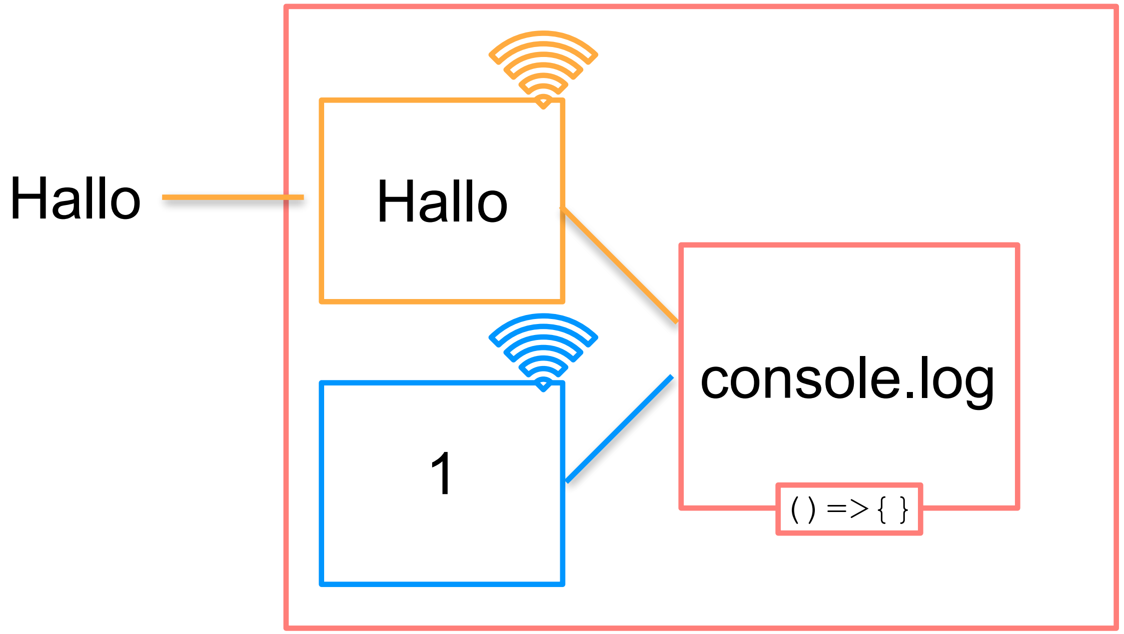
Effect: What - side effects, listen to signal / computed



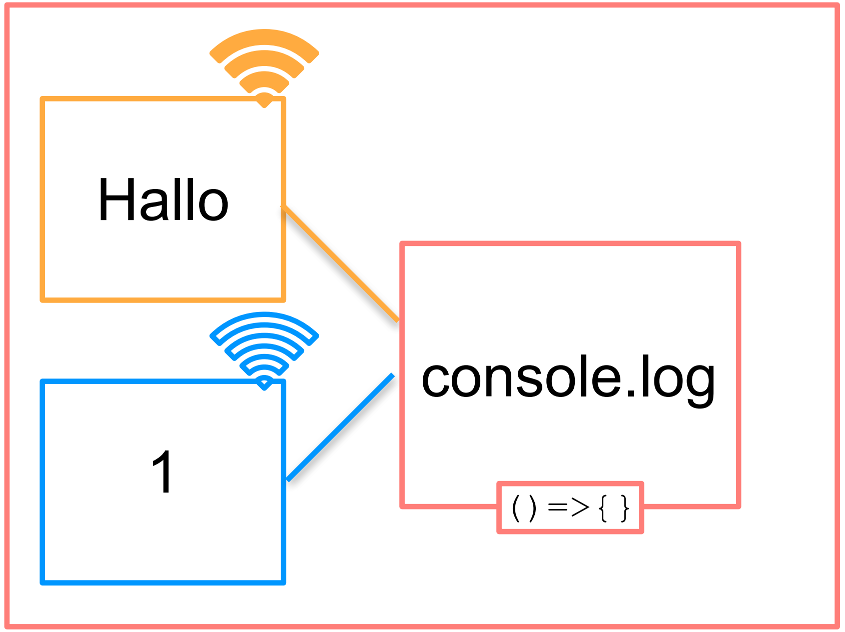
Effect: What - side effects



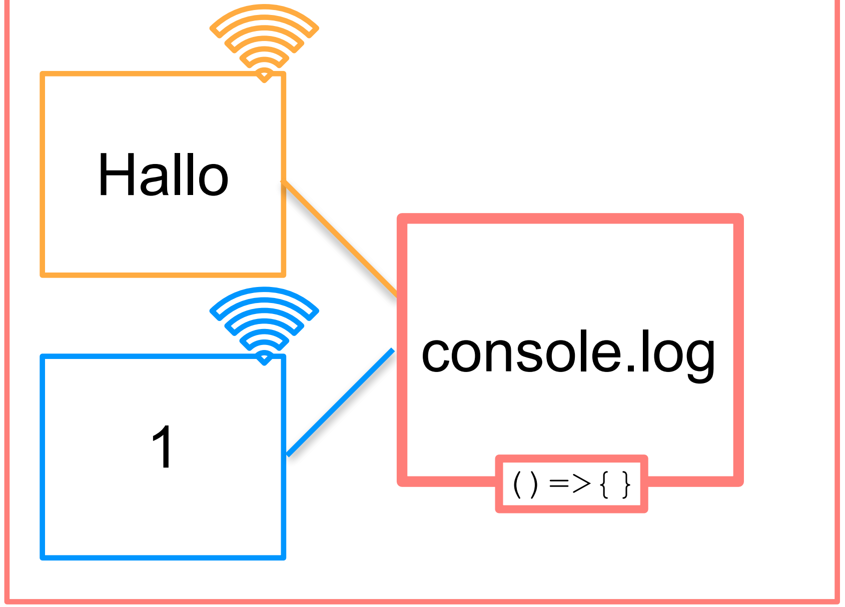
Effect: What - when any signal updates



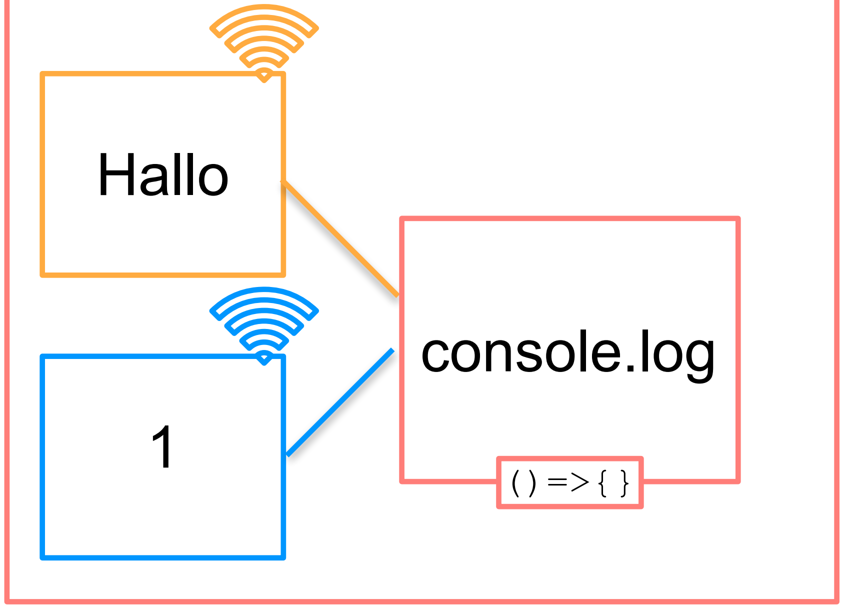
Effect: What - when any signal updates



Effect: What - The operation runs again



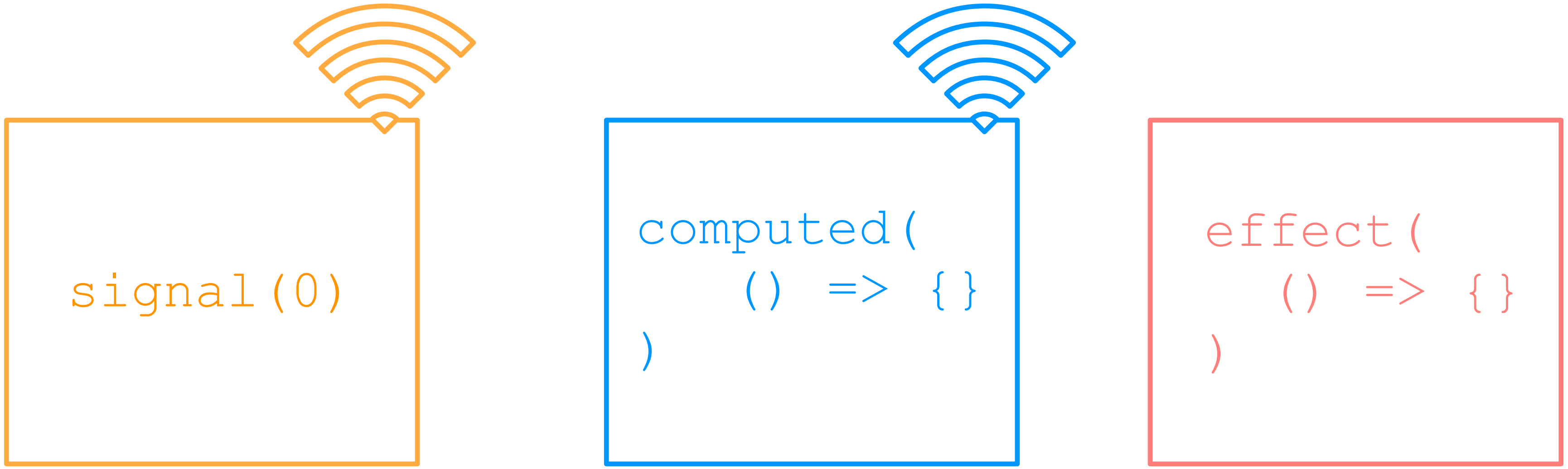
Effect: What - waits for updates



Effects Demo

RECONCEPT_

Signals: 3 primitives



```
signal(0)
```

```
computed(
    () => {}
)
```

```
effect(
    () => {}
)
```

Holds a value

can be updated

lazy

run operations

eager

no return

Signals: Notes

Signals are writable

Computed & effects can only read

Can’t write to signals in computed & effects*

 * ’allowSignalWrites’ exists for effects, but be careful

Signals: Notes

Signals & computed can be created anywhere

Effect needs an injection context

- So it can be cleaned up

Signals - SignalOptions

signal

- equal

computed

- equal

effect

- Injector
- allowSignalWrites
- manualCleanup

- onCleanup

More Signals Demo

Signals & RxJS

Interoperability

Signals & RxJS

toSignal

- Convert RxJS to signal
- Injection context
- Immediately subscribes

toObservable

- convert signal to RxJS
- Immediately emits

Signals Demo

RECONCEPT_

Future

Decoupling data from DOM structure

- Means the DOM is rendered when data is stable
- Technically: the template becomes a side effect of the data
- no more ExpressionChangedAfterItHasBeenChecked error!

Future - Angular 17

Signal Components

Can have fully Zoneless application

Will update specific ‘views’ in the application, no top down change detection

@inputs / outputs as signals

New lifecycle method registration instead of methods in the class

Future - Signal Based Component

```
bootstrapApplication( noZoneJS() )

@Component({
  signals: true,
  template: `
     {{ name() }}
  `
})
export class SignalComponent {
  name = signal('hello');
}
```

Future - Inputs

```
@Component({
  signals: true,
  selector: 'user-profile',
  template: `
    <h2>Hello {{name()}}!</h2>`,
})
export class InputSignalComponent {
  // needs initial value
  // is also readonly
  name = input('test');
}
```

All template reads MUST be done with Signals!!

Future - Output

```
@Component({
  signals: true,
  selector: 'user-profile',
  template: `
    <button (click)=“save.emit()”>Save</button>`,
})
export class OutputSignalComponent {
  // works basically the same
  save = output<string>();
}
```

Future - Signal Based Component

```
@Component({
  signals: true,
  selector: ‘is-admin',
  template: `
    <h2>isAdmin</h2>
    <input type="checkbox"(change)="update($event)">`,
})
export class TwoWayBindingComponent {
  checked = model(false);

  update(newValue: boolean) {
    // model is writeable
    this.checked.set(newValue)
  }
}
```

Future - Signal Based Component

```
@Component({
  signals: true,
  selector: 'parent',
  template: `
    <!-- we pass in the Signal, no "()" -->
    <is-admin [(checked)]="isAdmin" />`,
})
export class TwoWayBindingComponent {
  isAdmin = signal(false);

  constructor() {
    effect(() => {
      // effect is run when isAdmin signal changes
      const isAdmin = this.isAdmin();
    })
  }
}
```

Future - LifeCycle methods

ngDoCheck

ngOnChanges

ngOnInit

ngAfterViewInit

ngAfterViewChecked

ngAfterContentInit

ngAfterContentChecked

ngOnDestroy

Future - LifeCycle methods

~~ngDoCheck~~

~~ngOnChanges~~

ngOnInit

~~ngAfterViewInit~~

~~ngAfterViewChecked~~

~~ngAfterContentInit~~

~~ngAfterContentChecked~~

ngOnDestroy

Future - LifeCycle methods

ngDoCheck	->	effect()
ngOnChanges	->	computed()

Future - LifeCycle methods

ngAfterViewInit	perform action after rendering	afterRender() / afterNextRender()
ngAfterContentInit	do something with content	contentChild() / contentChildren()
ngAfterContentChecked	do something with content	contentChild() / contentChildren()
ngAfterViewChecked	do something with view	viewChild() / viewChildren()

Future - Signal Based Component

```
@Component({
   signals: true,
   template: `<some-component/>`
})
export class LifecycleComponent {
  name = input('');
  someComponent = viewChild(SomeComponent);

  constructor() {

    afterRender(() => {
      // After the DOM of *all* components has been fully rendered.
    });

    afterNextRender(() => {
      // Same as afterRender, but only runs once.
    });

    afterRenderEffect(() => {
      // Same as afterRender in terms of timing,
      // but runs whenever the signals which it reads have changed.
      console.log(`DOM was updated due to '${this.name()}'`);
    });

  }
}
```

More Angular 16

Application

- DestroyRef
  - takeUntilDestroyed()
- Route @Inputs
- self closing tags (15)

Building

- No ngcc
- esbuild-dev-server
- TypeScript 5.0
- migrate to standalone

SSR

- Non-destructive
- hydration support

Resources

To Read

Start RFC discussion

Complecte RFC Discussion

Information AMA with Alex Rickabaugh

Signals FAQ by Sander Elias

Signals By Manfred Steyer

Deepdive presentation about Angular Signals

To See

The talk that started it all

Angular Team going through the RFC

General talk about Future of Angular

Talk about how Signals work in detail

Talk about Angular & Signals in general

Stream about Signals & More (5h+)

Signals with Manfred Steyer