Manage RxJs Subscriptions without Subscribing in Angular

Angular Hackday Melbourne June 2020

Join the Conversation
#AngularHackdayMelbourne @geo_ge

Gabriel George

Senior Software Architect at SSW

@geo_ge

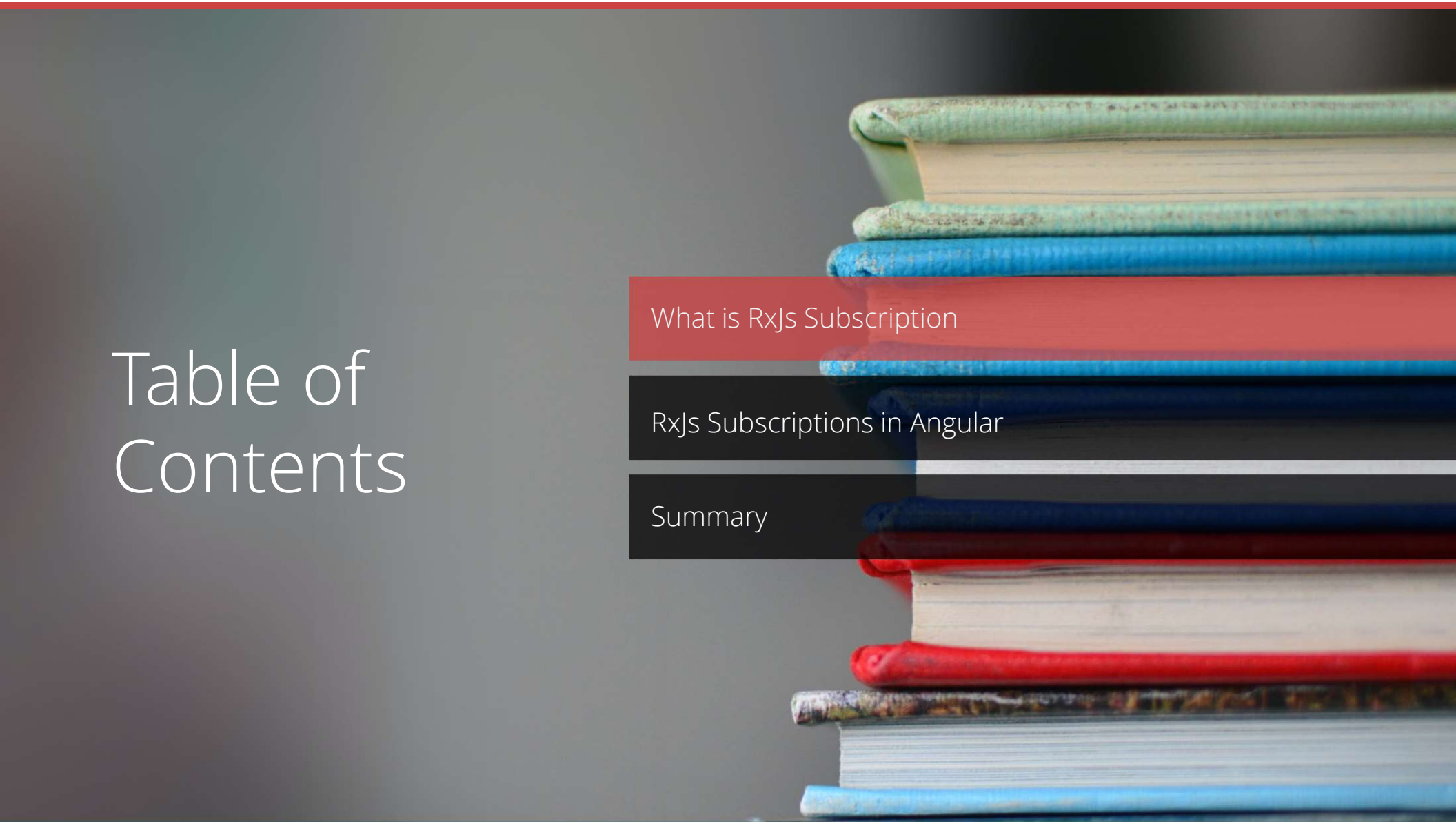
gabe@devzilla.io

devzilla.io

- Loves coffee! I'm nicer after coffee
- Currently playing Final Fantasy XIV on PC
- Used to want to be a doctor

Join the Conversation #AngularHackdayMelbourne @geo_ge

Table of Contents



What is Subscription?

Observable

> A representation of any set of values over any amount of time

`subscribe()`

> Invokes an execution of an Observable and registers Observer handlers for notifications it will emit

According to: https://rxjs-dev.firebaseapp.com/api/index/class/Observable

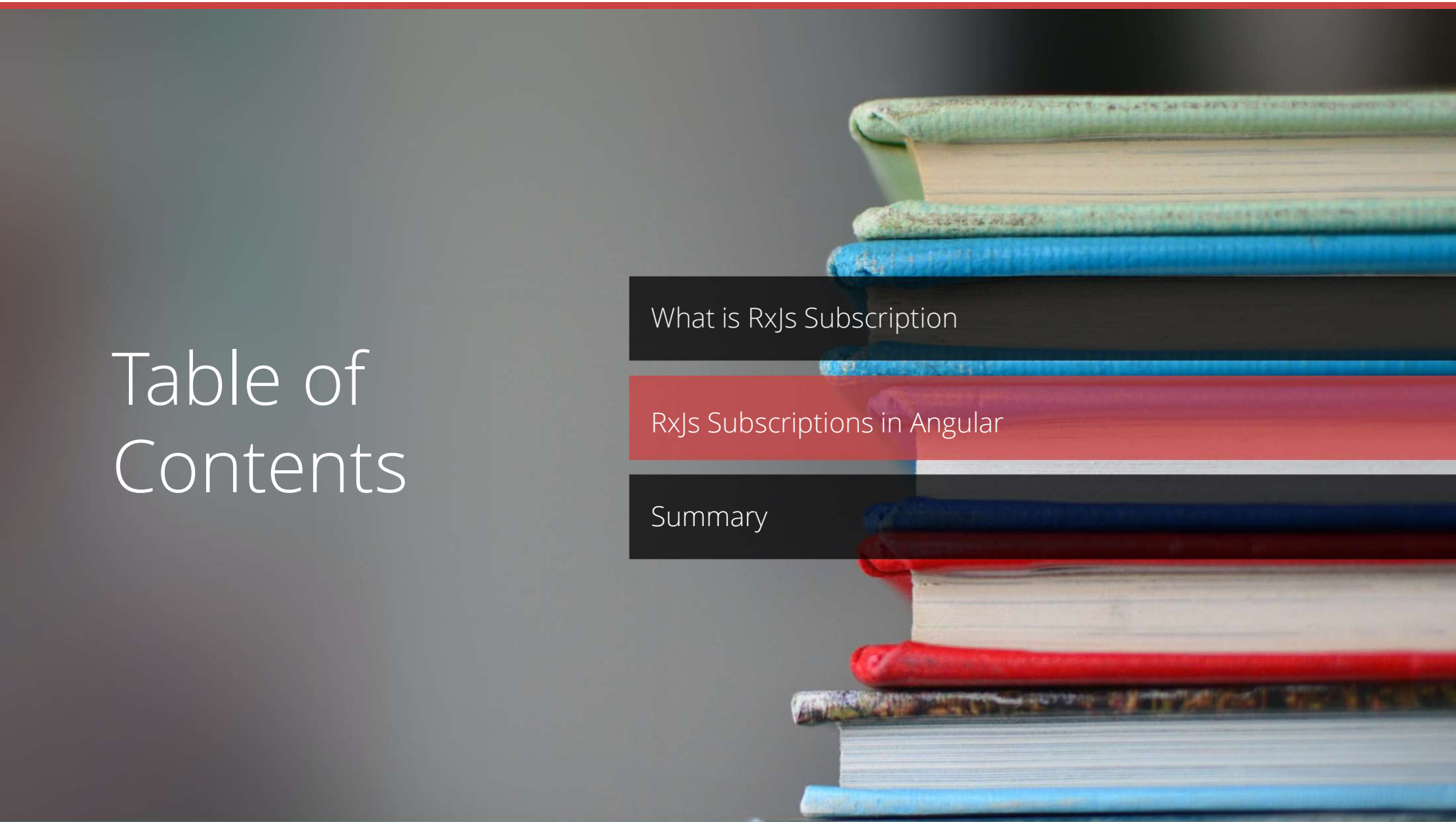
What is Subscription?

Subscription

- Is a disposable resource, usually the execution of an Observable
- Has one important method, `unsubscribe`
  - Disposes the resource held by the subscription

According to: https://rxjs-dev.firebaseapp.com/guide/subscription

Table of Contents



Where do we use Subscriptions?

Simplest use case:

```
this.httpClient
    .get('https://jsonplaceholder.typicode.com/todos')
    .subscribe((todos) => {
      ...
    });
```

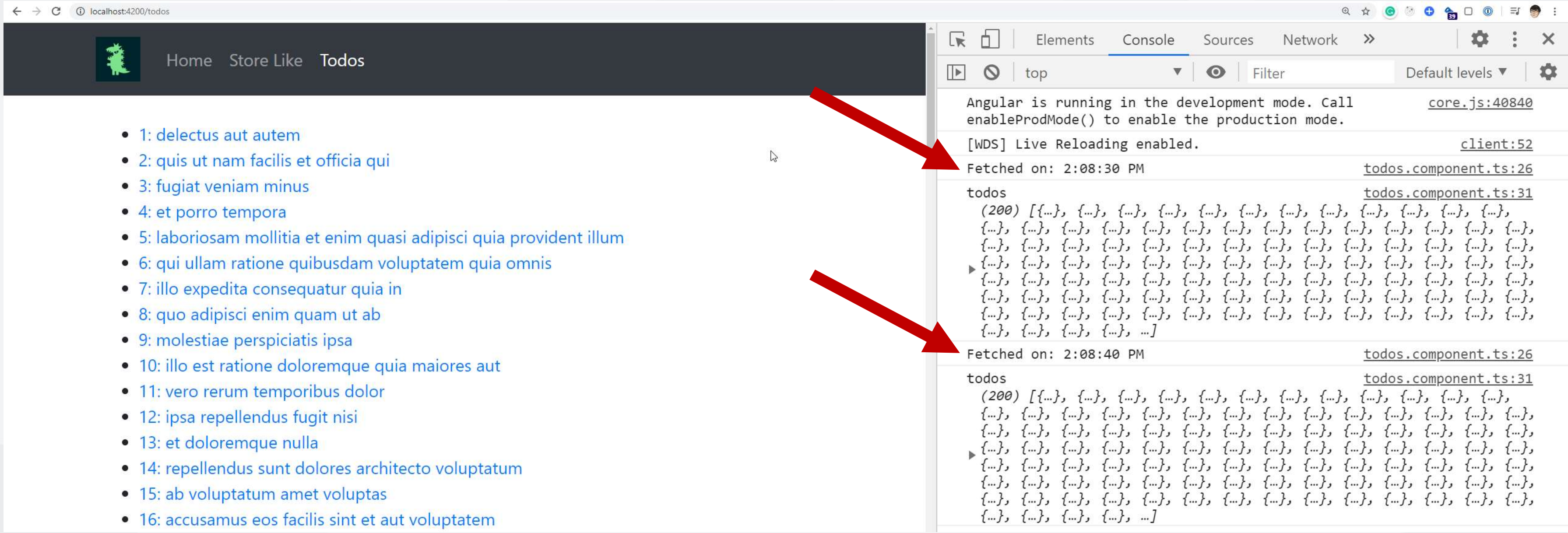
Where do we use Subscriptions?

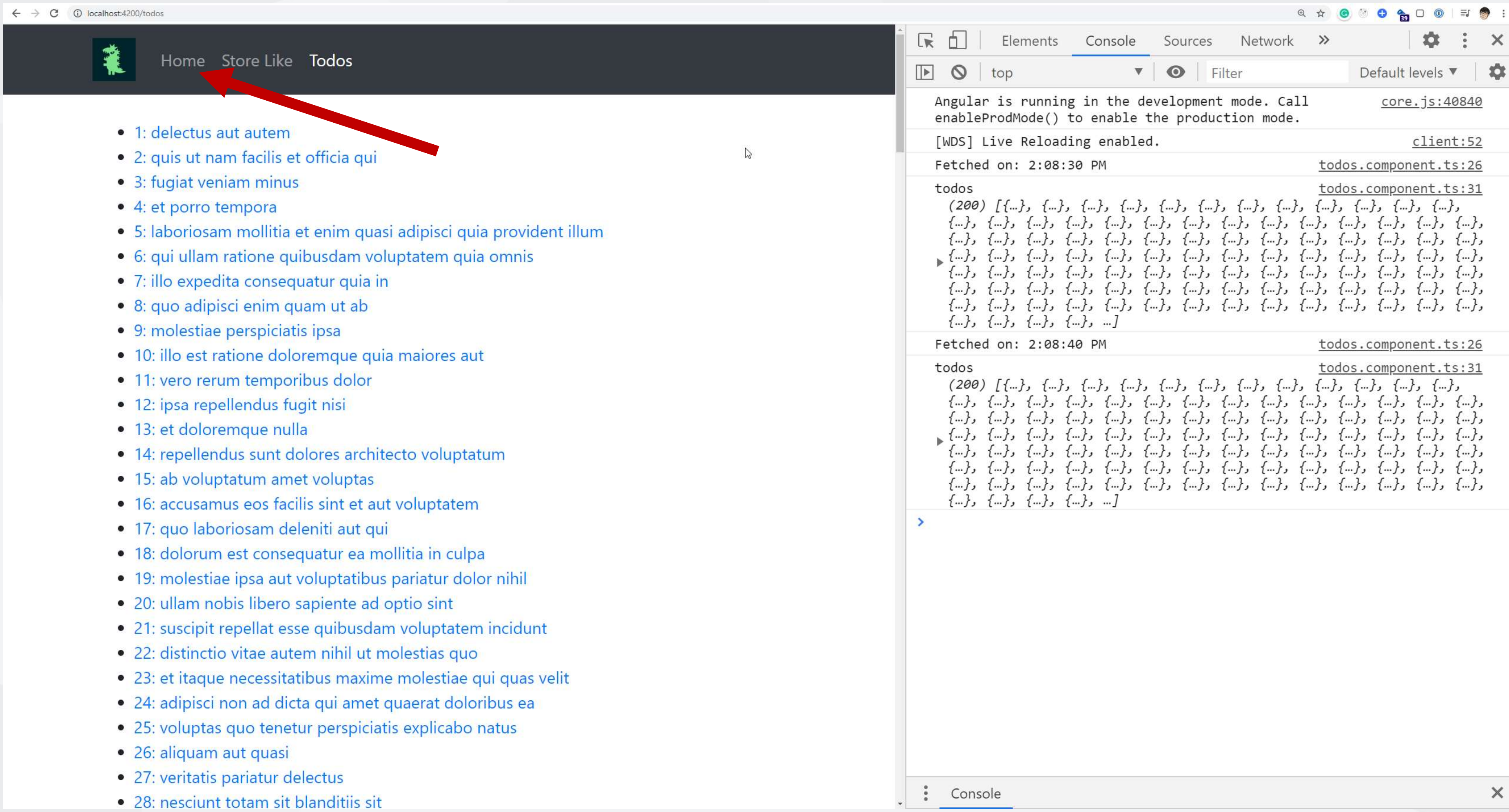
Wanting to always get the latest data every 10 seconds:

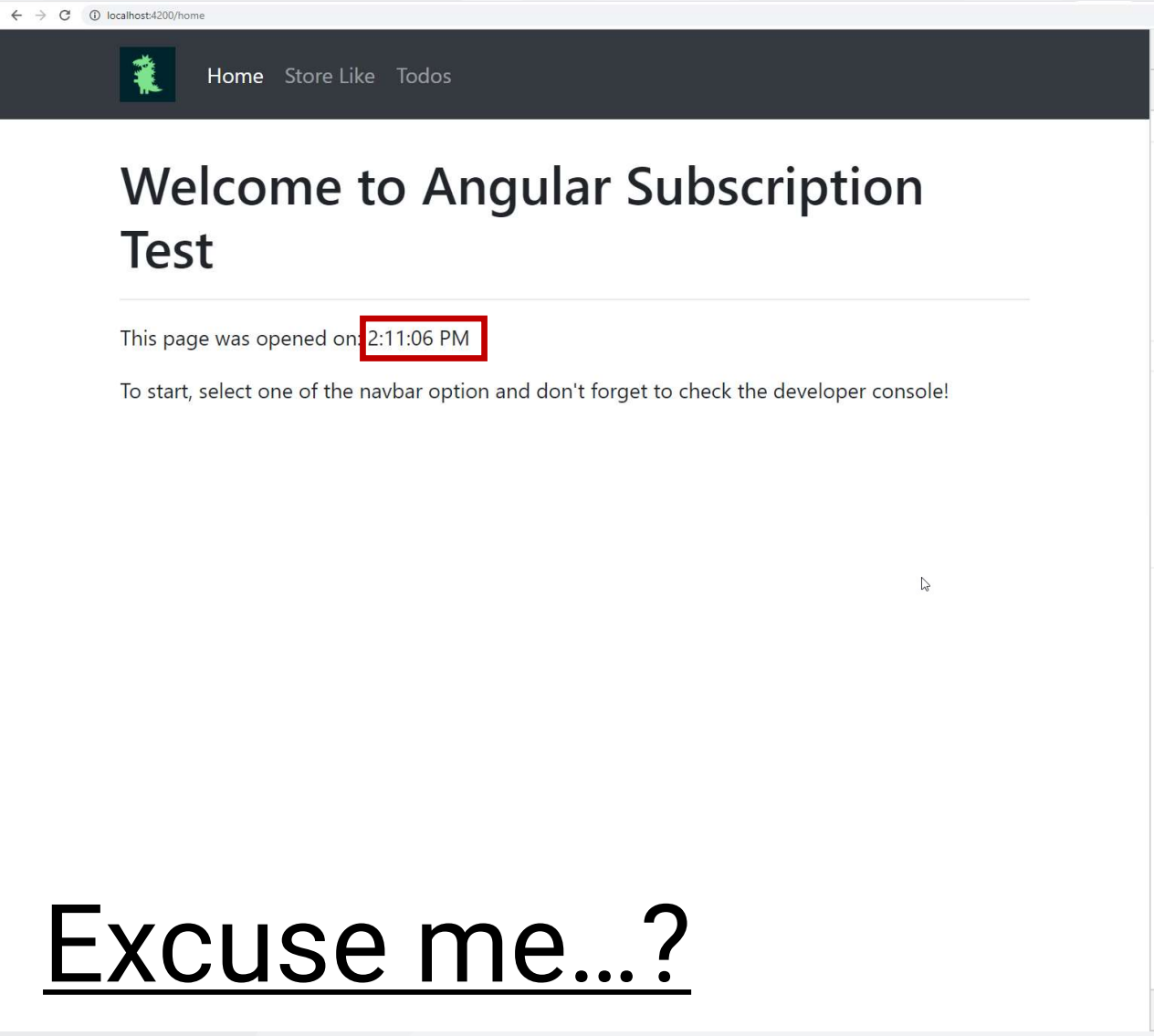
```
interval(10000).subscribe(() => {
      this.httpClient
        .get('https://jsonplaceholder.typicode.com/todos')
        .subscribe((todos) => {
          ...
        });
    });
```

Where do we use Subscriptions?

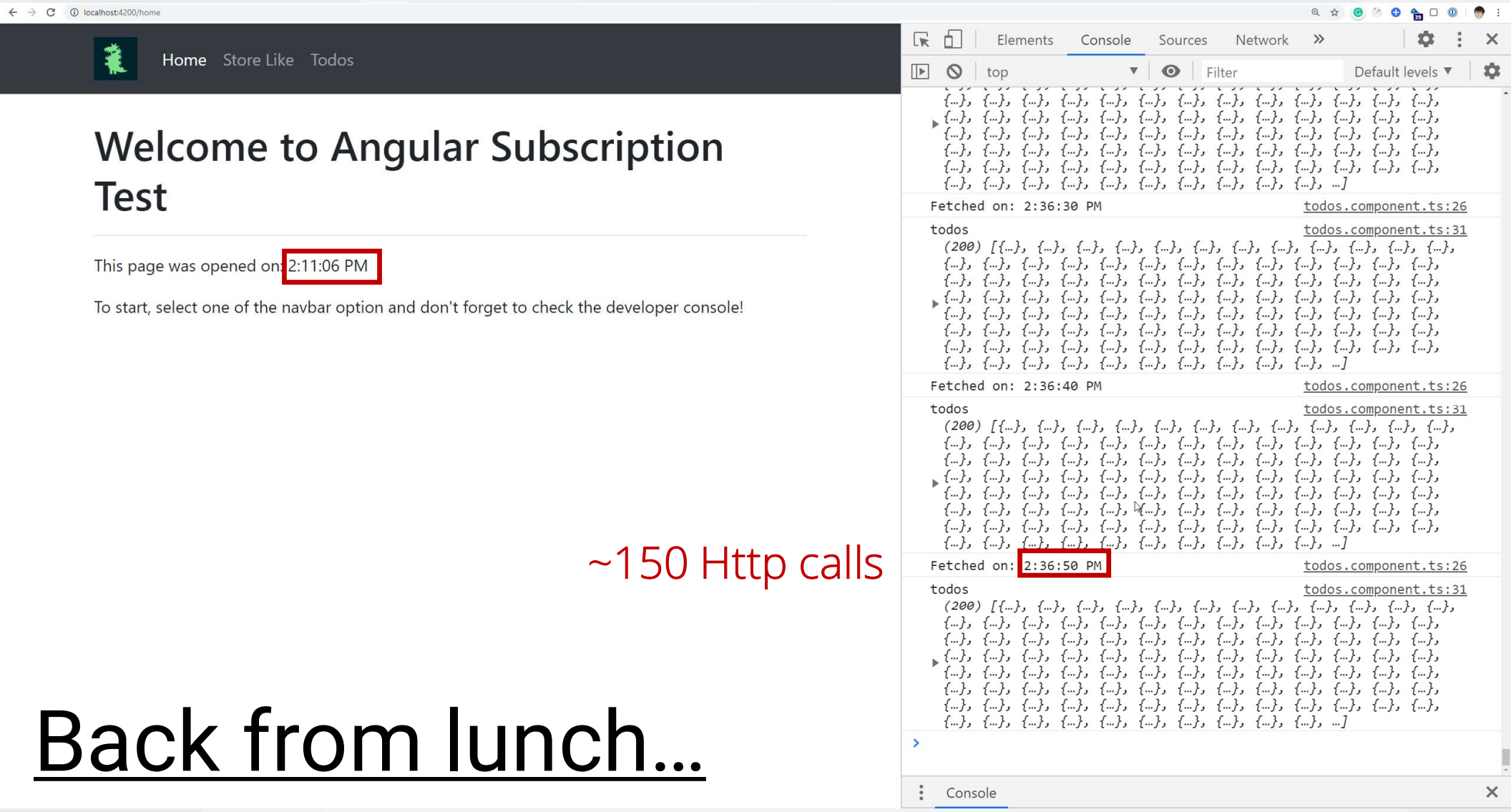
Wanting to always get the latest data every 10 seconds:




localhost:4200/todos
Home
Store Like
Todos
1: delectus aut autem
2: quis ut nam facilis et officia qui
3: fugiat veniam minus
4: et porro tempora
5: laboriosam mollitia et enim quasi adipisci quia provident illum
6: qui ullam ratione quibusdam voluptatem quia omnis
7: illo expedita consequatur quia in
8: quo adipisci enim quam ut ab
9: molestiae perspiciatis ipsa
10: illo est ratione doloremque quia maiores aut
11: vero rerum temporibus dolor
12: ipsa repellendus fugit nisi
13: et doloremque nulla
14: repellendus sunt dolores architecto voluptatum
15: ab voluptatum amet voluptas
16: accusamus eos facilis sint et aut voluptatem
17: quo laboriosam deleniti aut qui
18: dolorum est consequatur ea mollitia in culpa
19: molestiae ipsa aut voluptatibus pariatur dolor nihil
20: ullam nobis libero sapiente ad optio sint
21: suscipit repellat esse quibusdam voluptatem incidunt
22: distinctio vitae autem nihil ut molestias quo
23: et itaque necessitatibus maxime molestiae qui quas velit
24: adipisci non ad dicta qui amet quaerat doloribus ea
25: voluptas quo tenetur perspiciatis explicabo natus
26: aliquam aut quasi
27: veritatis pariatur delectus
28: nesciunt totam sit blanditiis sit
Elements
Console
Sources
Network
top
Filter
Default levels
Angular is running in the development mode. Call enableProdMode() to enable the production mode.
core.js:40840
[WDS] Live Reloading enabled.
client:52
Fetched on: 2:08:30 PM
todos.component.ts:26
todos
todos.component.ts:31
(200) [{…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…},
Fetched on: 2:08:40 PM
todos.component.ts:26
todos
todos.component.ts:31
(200) [{…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…},
Console


localhost:4200/home
Home
Store Like
Todos
Welcome to Angular Subscription Test
This page was opened on: 2:11:06 PM
To start, select one of the navbar option and don't forget to check the developer console!
Elements
Console
Sources
Network
top
Filter
Default levels
Fetched on: 2:11:10 PM
todos.component.ts:26
todos
todos.component.ts:31
(200) [{…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, …]
Fetched on: 2:11:20 PM
todos.component.ts:26
todos
todos.component.ts:31
(200) [{…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, …]
Console

Excuse me...?



~150 Http calls

Back from lunch...

What happened?

- I am no longer viewing my Todos page
- My previous subscription is still running
- I expect the subscription to only run when I am viewing the page
- When I open my Todos page again, the subscription is doubling!

Pain points

- Angular does not terminate our subscriptions even when the component gets destroyed
- In a complex Angular application:
  - Potentially cause unexpected side-effects
  - Unnecessary resource consumption
  - Who knows what else might happen...

Looks like plumbing issue!

How can we fix this?

1. Unsubscribe all Subscriptions
2. Completing RxJs pipe
   - take()
   - takeWhile()
   - takeUntil()
3. Use Angular async pipe

Unsubscribe all Subscriptions

Root cause of all unwanted side-effects are unterminated subscriptions.

It sounds the most sensible thing to do!

Unsubscribe all Subscriptions

Root cause of all unwanted side-effects are unterminated subscriptions.

It sounds the most sensible thing to do!

Unsubscribe: Single Subscription

```
ngOnInit() {
  // Assign the subscription to a variable
  this.subscription = interval(10000).subscribe(() => {
    ...
  });
}

ngOnDestroy() {
  // Manually unsubscribe subscription on ngDestroy life-cycle
  this.subscription.unsubscribe();
}
```

Unsubscribe: Subscriptions in array

```
ngOnInit() {
  // Push every subscriptions to an array of subscriptions
  this.subscriptions.push(
    interval(10000).subscribe(() => {
      ...
    })
  );
}

ngOnDestroy() {
  // Loop on every subscriptions and unsubscribe
  this.subscriptions.forEach((subscription) => {
    subscription.unsubscribe();
  });
}
```

Unsubscribe...

- ✅ Does the job
- ✅ Easy to understand

- ❌ Tedious job
- ❌ Can't unsubscribe if you lose the reference
- ❌ Unsubscribing completed subscription will throw an error
- ❌ It's easy to forget to unsubscribe

Completing RxJs pipe

RxJs has various operators that can complete a stream

- take()
- takeWhile()
- takeUntil()
- ...

Completing RxJs pipe: take()

```
ngOnInit() {
  interval(10000)
    // take operator only takes X count and then completes
    .pipe(take(10))
    .subscribe((x) => {
      ...
    });
}
```

Completing RxJs pipe: takeWhile()

```
private _componentExist = true;

ngOnInit() {
  interval(10000)
    // takeWhile pipe will accept stream as long as the value returns true.
    // When the value is false, it will terminate the stream
    .pipe(takeWhile(() => this._componentExist))
    .subscribe(() => {
      ...
    });
}

ngOnDestroy() {
  this._componentExist = false;
}
```

Completing RxJs pipe: takeUntil()

```
private ngDestroy$ = new Subject();

ngOnInit() {
  interval(10000)
    // takeUntil will all accept value until the Observable inside takeUntil emits value
    .pipe(takeUntil(this.ngDestroy$))
    .subscribe((x) => {
      ...
    });
}

ngOnDestroy() {
  // Emit value to make all takeUntil gets triggered
  this.ngDestroy$.next();
  // Then complete its own subscription to make sure there is no loose end
  this.ngDestroy$.complete();
}
```

Completing RxJs pipe

- ✅ No need to unsubscribe manually
- ✅ Can be unsubscribed whenever needed
- ✅ No need to store the subscription's reference

- ❌ Slightly more learning curve
- ❌ Need to manually set the condition when to complete

Use Angular async pipe

Angular async pipe will

- Subscribe automatically when component is initialised
- Unsubscribe automatically when the component gets destroyed.

Use Angular async pipe

TS

```ts
data$: Observable<string>;
ngOnInit() {
  this.data$ = interval(10000).pipe(
    ...
  );
}
```

HTML

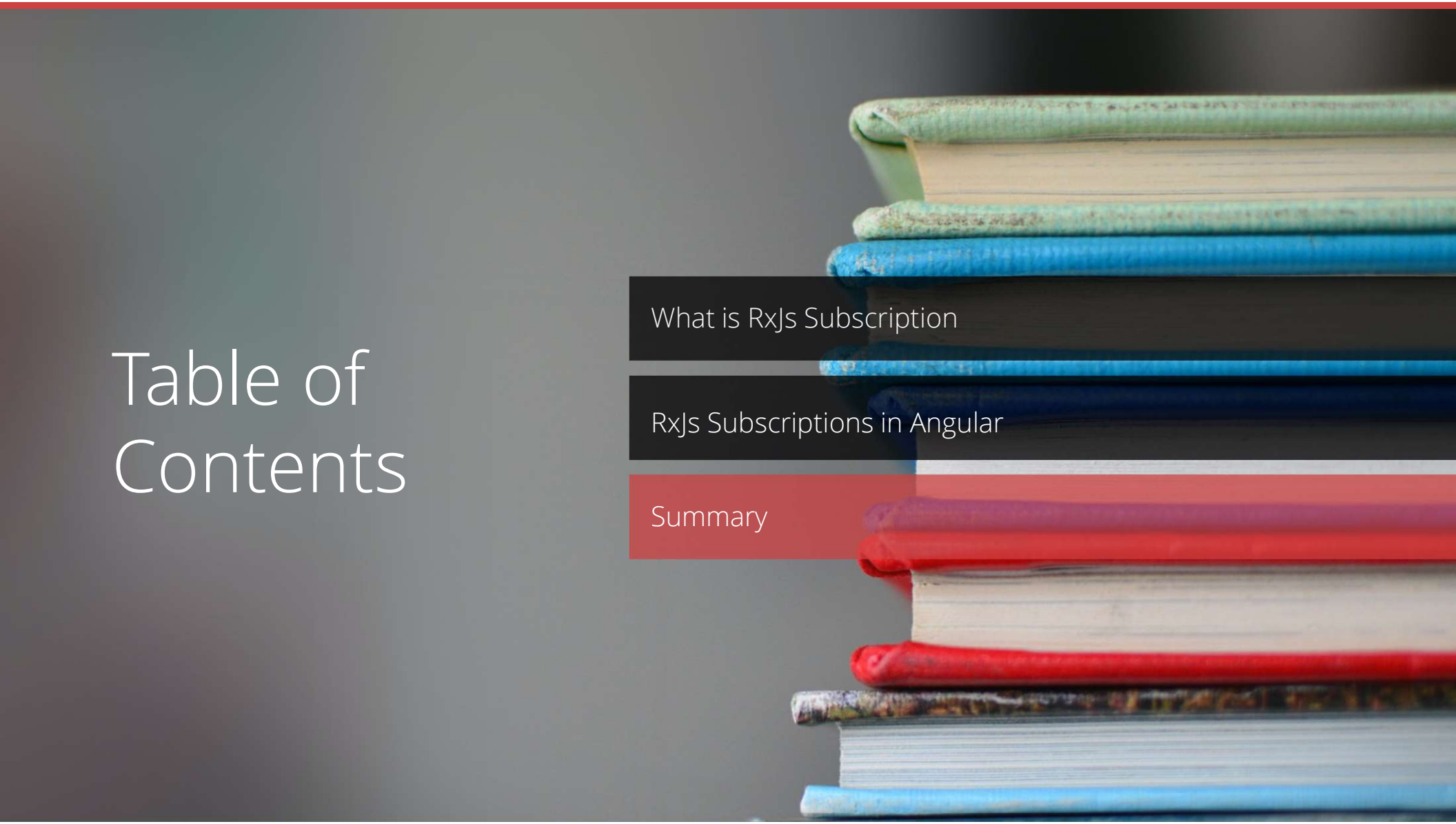
```html
<p *ngIf="data$ | async as data">
  {{ data }}
</p>
```

Use Angular async pipe

- ✅ Auto-subscribe on initialisation
- ✅ Auto-unsubscribe on destroy
- ✅ Smallest code footprint
- ✅ No `subscribe` in component.ts

- ❌ Require more RxJs knowledge to fully leverage the pipe
- ❌ Tightly coupled to Angular life-cycle (for better or worse)

Table of Contents



Summary

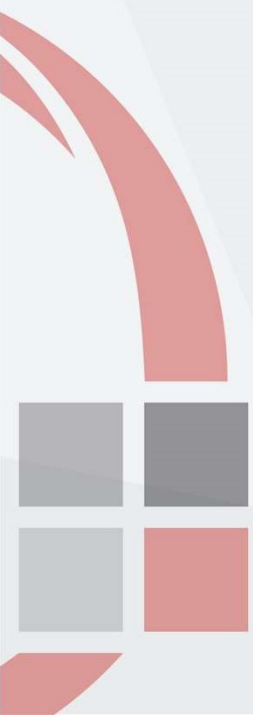
We can terminate subscriptions by

1. Manually unsubscribe all Subscriptions
2. Completing RxJs pipe
    - take()
    - takeWhile()
    - takeUntil()
3. Use Angular async pipe

Summary

By terminating all subscriptions

- Prevent unwanted side-effects
- Remove potential resource wasting



Thank you!

Tweet me on @geo_ge

Will be blogged on: devzilla.io