State management

- Il existe des libraires dédiées au state management pour Angular (NgRx, NGXS), mais elles ne sont pas obligatoires
- Dans le futur, state management avec des Signals ?

Gestion d’erreur avec les opérateurs

- retry(n) va tenter de s’abonner jusqu’à n fois à un observable si celui ci emet une erreur

```
retry<T>(configOrCount: number | RetryConfig = Infinity): MonoTypeOperatorFunction<T>
```

- catchError() remplace l’observable source par un autre en cas d’erreur
- catchError() prend un callback en paramètre
- Ce callback prend deux paramètres, l’erreur à "catch", et optionellement l’observable source, et renvoie un nouver observable

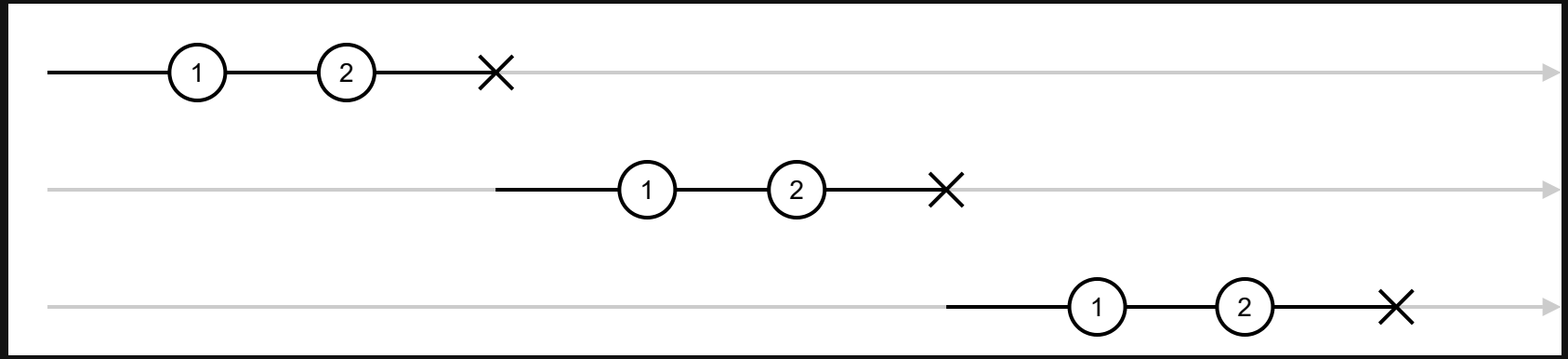
```
catchError<T, O extends ObservableInput<any>>(selector): OperatorFunction<T, T | ObservedValueOf<O>>
selector: (err: any, caught: Observable<T>) => O
```

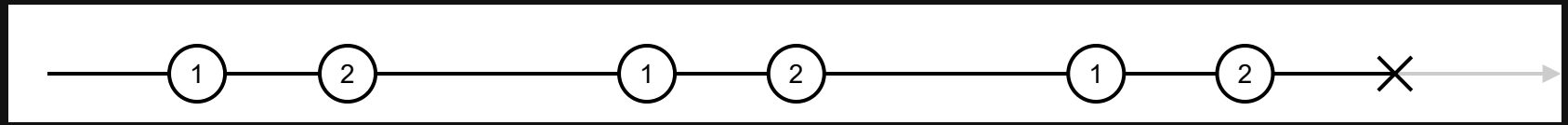
(penser a catch() en Java par exemple)

- Exemple, je considère un observable obs$ qui émet 1, 2 puis une erreur

- L'opérateur retry(2) va se réabonner jusqu'a deux fois lors de l'émission de l'erreur

```
obs$.pipe(retry(2))
```

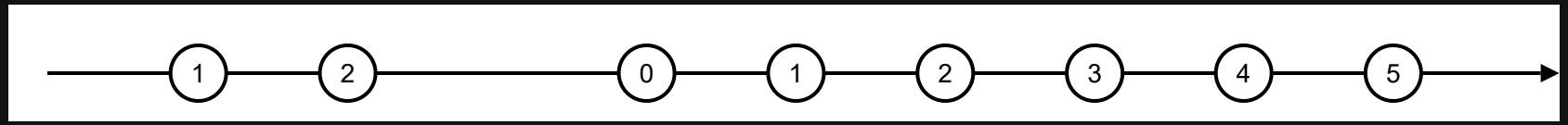




- A partir du même observable que la slide précédente

- Avec l'opérateur catchError(), je substitue un autre observable lors d'une erreur

```
obs$.pipe(
   catchError(() ⇒ interval(1000))
)
```



- Dans cet exemple, lors de l'erreur de l'observable initial, je le remplace par interval(1000)

Retour sur HttpClient

Méthodes HTTP

- Les méthodes pour les requètes HTTP courantes sont également implémentées dans le service HTTPClient :
  - post() / POST
  - put() / PUT
  - delete() / DELETE
  - patch() / PATCH
  - head() / HEAD
  - options() / OPTIONS

POST / post()

- La requète post permet d'envoyer des données au serveur

```
_httpClient.post.post(url, body, options): Observable<Object>
```

- Comme pour la requète get(), le type de retour de la méthode est un observable, qui renvoie le contenu du body de la réponse du serveur, supposée au format JSON et convertie en objet

Typage

- Par défaut le client http de Angular retourne uniquement le body de la réponse de serveur, au format JSON convertie en object

- On peut utiliser les options dans les méthodes du client pour changer cela :

```
responseType?: 'json' (default) | 'arraybuffer' | 'blob' | 'text'
```

```
observe?: 'body' (default) | 'events' | 'response'
```

- Le type de retour de l'observable change avec les paramètres