**HttpClient**

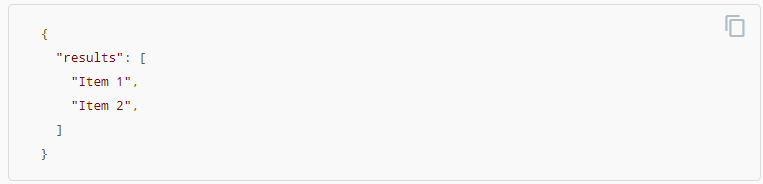
# Setup: installing the module

ก่อนที่จะใช้งาน HttpClient เราต้องทำการติดตั้ง HttpClientModule

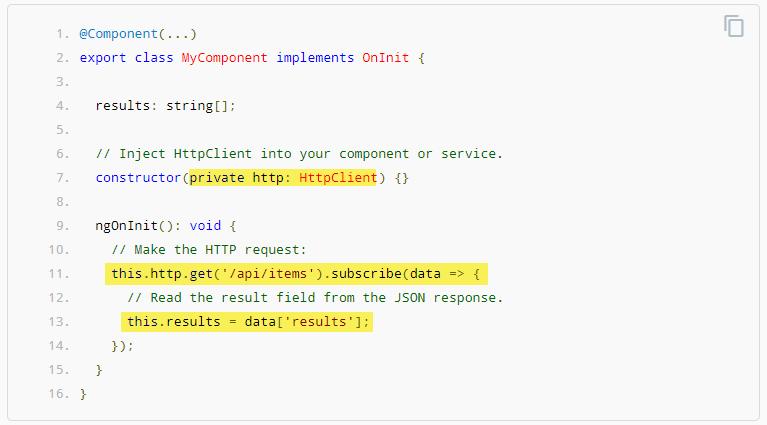


# Making a request for JSON data

Request แบบ JSON data จะเป็นวิธีที่ใช้บ่อยที่สุดเมื่อต้องการข้อมูลจาก Backendตัวอย่าง: ถ้าเรามี API ที่มีปลายทางเป็น lists items, /api/items, ที่จะ return เป็น JSON object

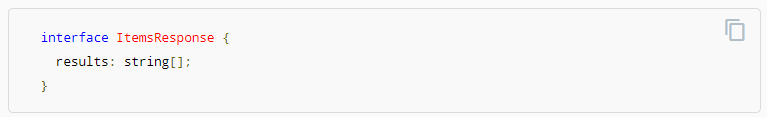


เราจะใช้ get() method ของ HttpClient เข้าถึงข้อมูลตรงนี้

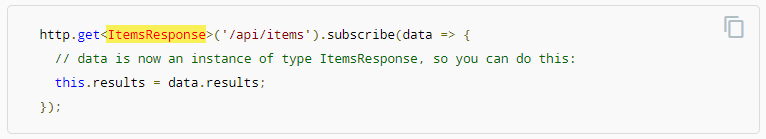


## Typechecking the response

เราสามารถบอก HttpClient ประเภทของ response เราสามารถสร้าง Interface และกำหนดรูปร่างที่ถูกต้องได้



แล้วก็เราทำการเรียก HttpClient.get โดยส่ง type parameter ไป



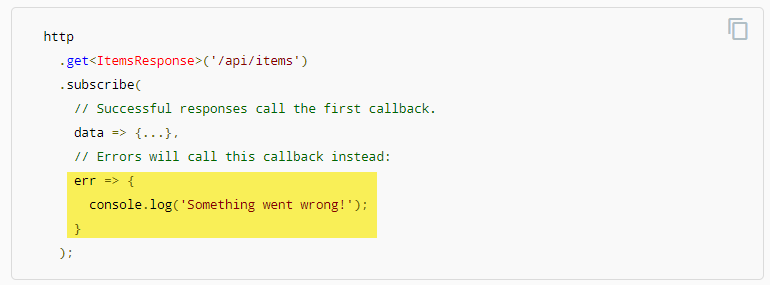
## Reading the full response

Response body ไม่ทำการ return ข้อมูลทั้งหมดที่เราต้องการ ในบางครั้ง server จะ return special headers หรือ status codes เพื่อแสดงบางเงื่อนไข และสิ่งที่จำเป็น ซึ่งเราสามารถกำหนดให้ HttpClient ตอบสนองแบบเต็มรูปแบบได้ เพียงแค่แทน body ด้วย observe option



## Error handing

อาจจะเกิดจาก request fails หรือ network มีปัญหา ซึ่ง HttpClientสามารถเพิ่ม error handler ใน .subscribe() ได้ เพื่อใช้จัดการ



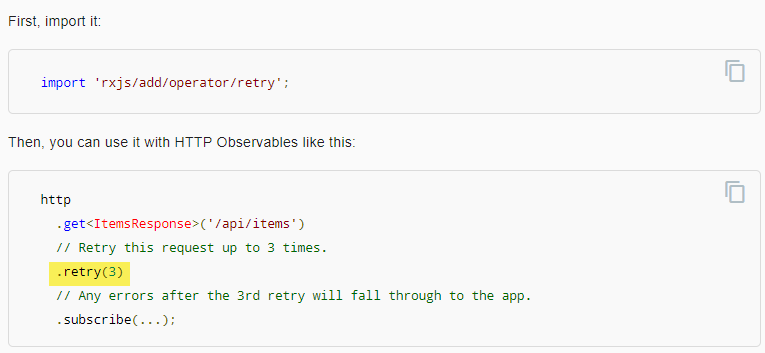
## Getting error details

เราสามารใช้ HttpErrorResponse เพื่อดูว่าเกิดไรขึ้นได้ อย่างเช่นถ้า backend return response code (404, 500, etc.)



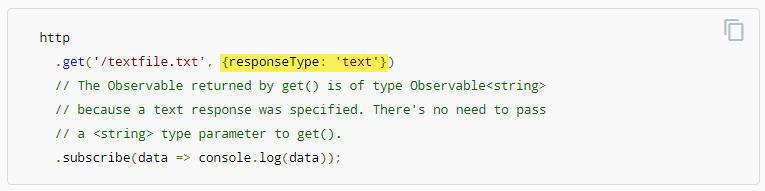
.retry()

RxJS มี operator ให้เรียกใช้ คือ .retry() ซึ่งใช้ในการ request ใหม่เมื่อพบ error



## Requesting non-JSON data

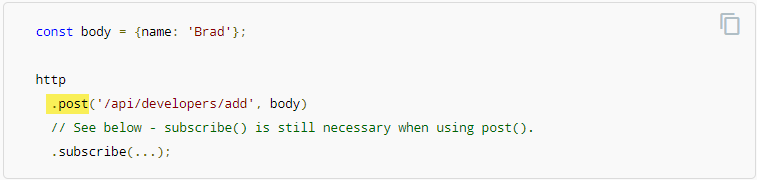
All APIs ไม่สามารถ return JSON data ได้ ในบางครั้งถ้าเราต้องการอ่านไฟล์ text จาก server เราก็สามารถกำหนด response type เป็น text ได้



# Sending data to the server

นอกเหนือจากการเรียกข้อมูลจาก server นั้นคือ HttpClient สามารถส่งข้อมูลให้ server ในรูปแบบต่างๆ

## Making a POST request





## Configuring other parts of the request



# Advanced usage

## Intercepting all requests or responses

Interceptors อยู่ระหว่าง application และ backend มันสามารถแปลง Request จาก application ก่อนส่งไปให้ server และสามารถแปลง Response จาก server กลับก่อนที่ application จะเห็น

#### Writing an interceptor



#### Providing your interceptor

การให้ NoopInterceptor ใช้งานได้ต้องประกาศใน providers ก่อน



multi: true คือสิ่งที่จำเป็นในการบอก Angular ว่า HTTP\_INTERCEPTORS คือ array of values ไม่ใช่ single value

#### Events

Observable ที่ return โดย Intercept และ HttpHandler.handle ไม่ใช่ Observable<HttpResponse<any>> แต่เป็น Observable<HttpEvent<any>> นั่นเป็นเพราะ interceptors ทำงาน lower level กว่า HttpClient interface

ใน single request สามารถ generate ได้หลายๆ events ทั้ง upload และ download events ความคืบหน้าต่างๆ

#### Ordering

เมื่อมีหลายๆ Interceptors ใน application ตัว Angular จะใช้พวกนั้นตามลำดับที่เราให้ไว้

#### Immutability

ถ้าต้องการเปลี่ยนแปลง request body เราจะต้อง copy request และ set new body ให้ด้วย clone()



#### Setting new headers

โดยทั่วไป เราจะใช้ Interceptors ตั้งค่า default header ในการตอบกลับ โดยตัวอย่างในนี้ เราสมมติว่า เรา injectable AuthService โดยสามารถใช้ authentication token ได้ เราจะเขียน interceptor ได้ดังนี้



และ shortcut สำหรับ set new headers



#### Logging

Interceptors สามารถประมวลผล request และ response พร้อมๆกัน พวกมันสามารถทำสิ่งต่างๆได้ เช่น เก็บ log หรือ time request ในตัวอย่าง Interceptor นี้ จะเป็นการใช้ console.log เพื่อแสดงระยะเวลการขอ request แต่ละครั้ง



Do() มันจะมีผลข้างเคียงกับ Observable แต่ไม่มีผลกับค่าใน stream

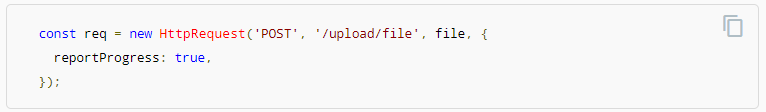
#### Caching

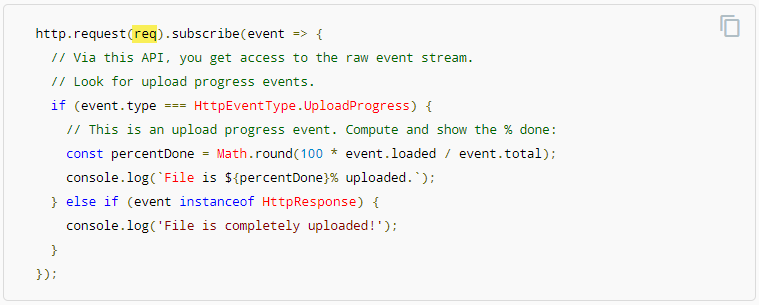
Interceptors สามารถ implement caching ได้ โดยตัวอย่าง จะสมมติว่า เรามีการเขียน HTTP cache ด้วย simple interface



Interceptor สามารถ apply cache นี้กับ request  


## Listening to progress events

เมื่อต้องการทำ request ด้วยการเปิด Progress event สิ่งแรกคือสร้าง instance ของ HttpRequest ด้วย reportProgress

จากนั้นทำการร้องขอผ่าน request() ของ HttpClient จะได้ผลลัพธ์เป็น Observable ของ events เช่นเดียวกับ Interceptors

# Security: XSRF Protection

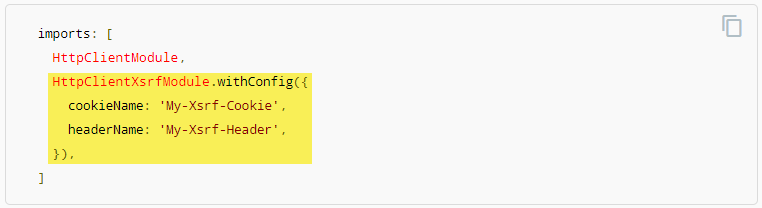
Cross-Site Request Forgery (XSRF) คือ เป็นเทคนิคการโจมตีเว็บไซต์ที่แฮ็คเกอร์ นิยมใช้มากขึ้นเรื่อยๆ ซึ่งเป็นการโจมตีที่ใช้ประโยชน์จากความเชื่อของเว็บไซต์ที่มีต่อข้อมูล Input และเบราเซอร์จากผู้ใช้งาน กล่าวคือ เหยื่อจะถูกหลอกให้กระทำการบางอย่างบนเว็บไซต์ปกติทั่วไป ที่ก่อให้เกิดประโยชน์ต่อแฮ็คเกอร์ในนามของตัวเหยื่อเอง โดยที่ตัวเหยื่อไม่ต้องการกระทำหรือกระทำไปโดยไม่รู้ตัว

การโจมตีแบบ CSRF จะใช้ “ตัวตน (Identity)” และ “สิทธิ์ (Privilege)” ของเหยื่อที่มีบนเว็บไซต์ ในการปลอมตัวเป็นเหยื่อและกระทำการหรือธุรกรรมไม่พึงประสงค์ แฮ็คเกอร์จะพยายามใช้ประโยชน์จากเหยื่อที่มี Login Cookies เก็บไว้ในเว็บเบราเซอร์

วิธีป้องกัน คือ ฝั่ง server เปลี่ยนจากการใช้ Cookies ที่ทำ Session-tracking มาเป็น Session Token ที่ถูกสร้างขึ้นแบบไดนามิกแทน ซึ่งช่วยให้แฮ็คเกอร์ขโมยเซสชันจากผู้ใช้ได้ยากยิ่งขึ้น

## Configuring custom cookie/header names

ถ้าต้องการ override ค่า default ของ XSRF cookie หรือ header ให้ใช้ HttpClientXsrfModule.withConfig()



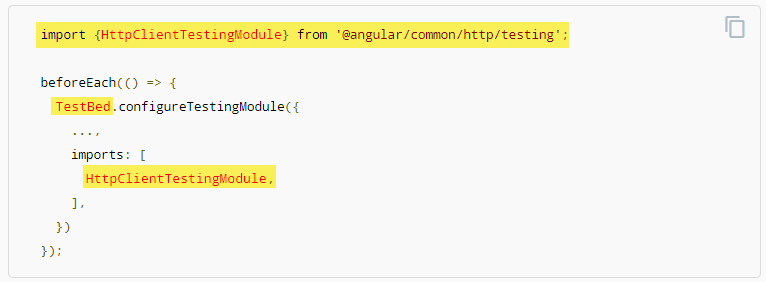
# Testing HTTP requests

## Mocking philosophy

Angular’s HTTP testing library คือ การออกแบบ pattern การทดสอบ โดยการ execute code และทำ request อันดับแรก และทดสอบว่ามีการทำ request หรือไม่ ยืนยัน request เหล่านั้น และทำการ flushing แต่ละ request สุดท้ายการทดสอบสามารถยืนยัน application ไม่ทำ requests ที่ไม่คาดคิด

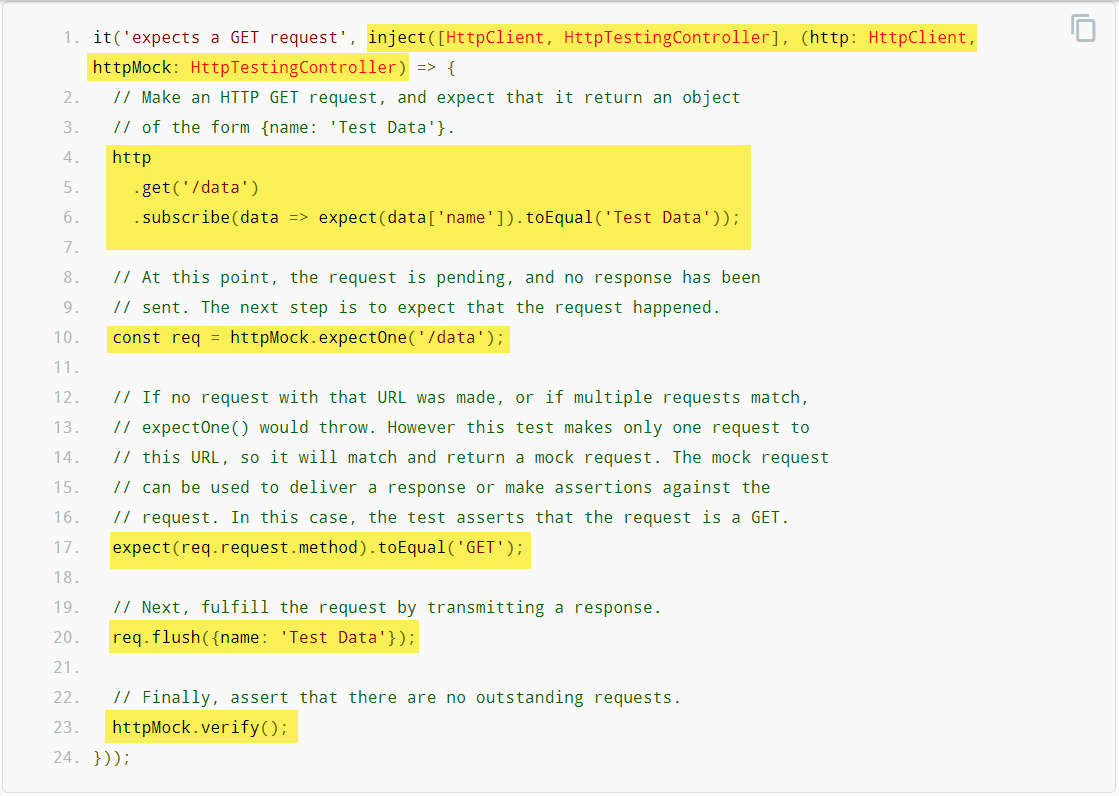
## Setup

การเริ่ม test request HttpClient เราต้อง import HttpClientTestingModule และเพิ่มลงใน TestBed

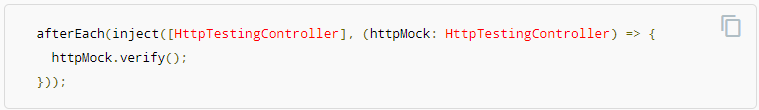


## Expecting and answering requests

หลังจากติดตั้ง mock ผ่านทาง Module แล้ว เราสามารถทดสอบโดยตัวอย่างจะทำการ Inject HttpClient เข้าไปใน test และเรียก HttpTestingController

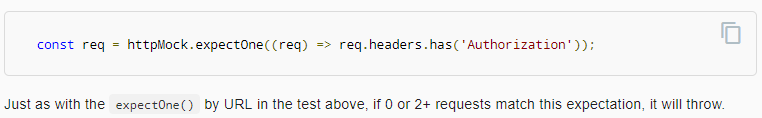


ในขั้นตอนสุดท้าย เราสามารถยืนยันว่าไม่เหลือ request แล้ว เราแค่ย้ายมันเข้าไปใน afterEach()



#### Custom request expectations

ถ้าการการจับคู่ URL ยังไม่เพียงพอ เราสามารถใช้ function ของเราได้ โดยในตัวอย่าง เราสามารถมองหา request ที่มี Authorization ใน header



#### Handling more than one request

ถ้าต้องการ respond ที่ request ซ้ำๆ ในการ test เราสามารถใช้ match() แทน expectOne()

