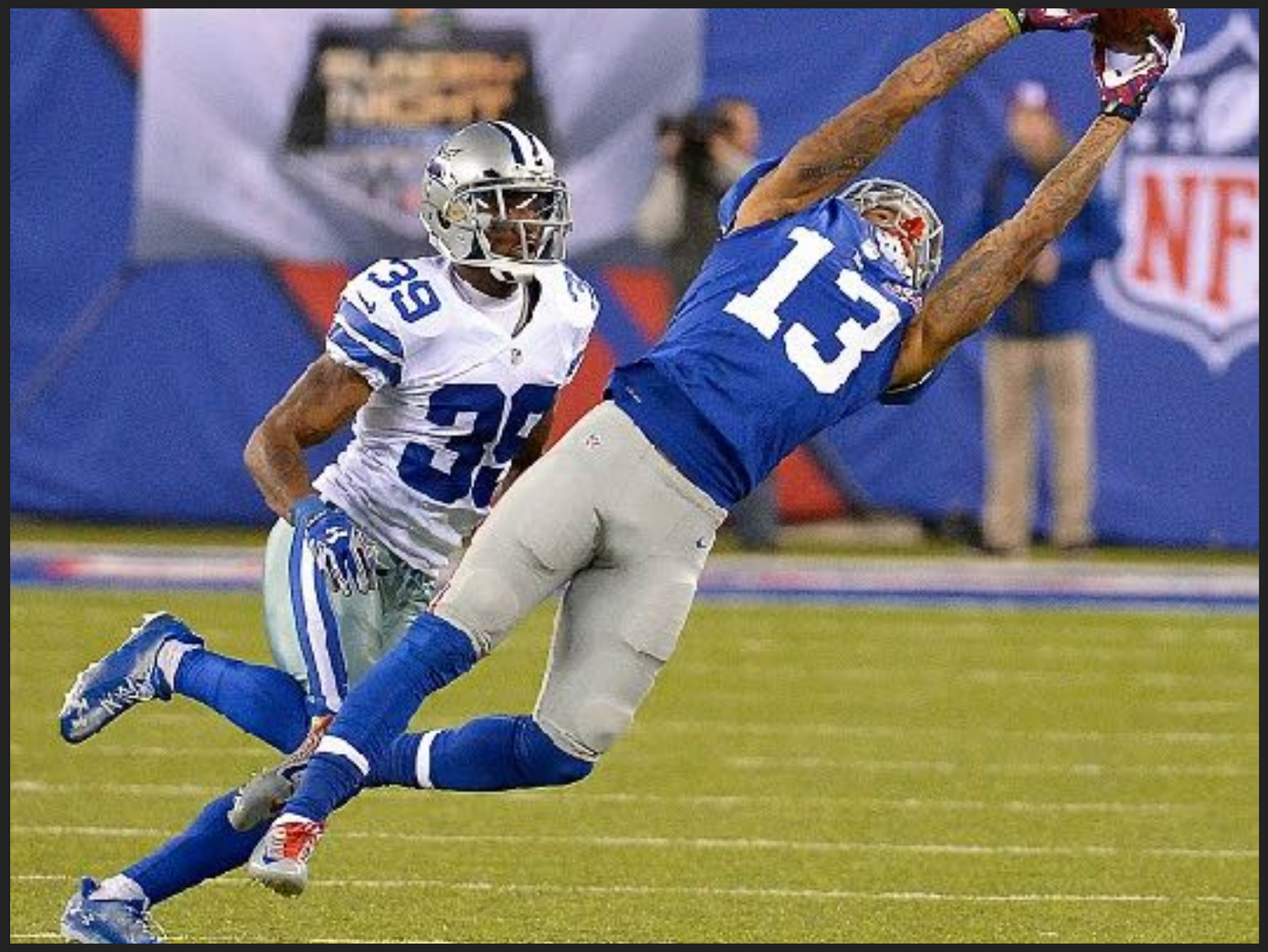
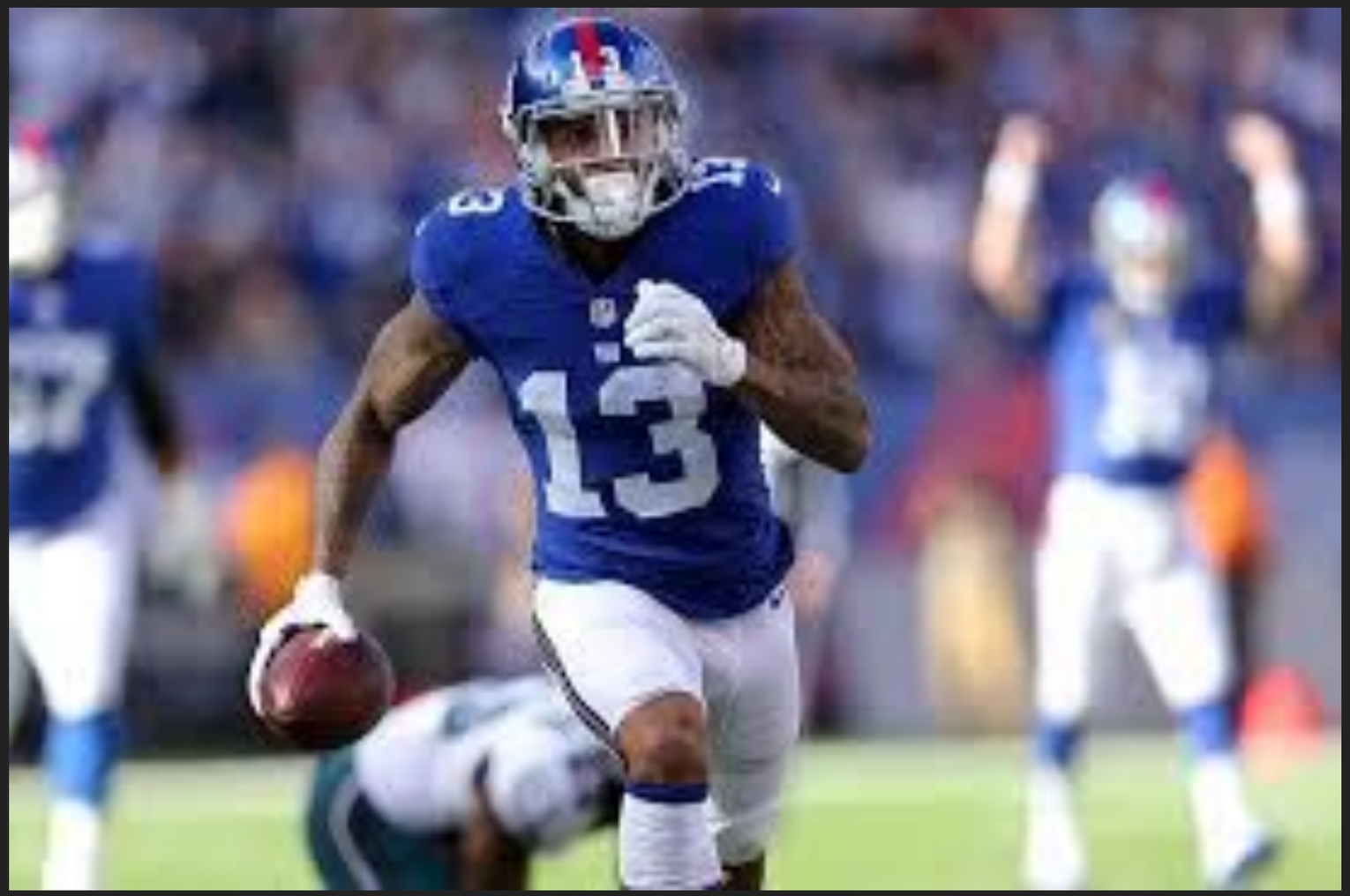


Introduction to

INTERCEPTORS

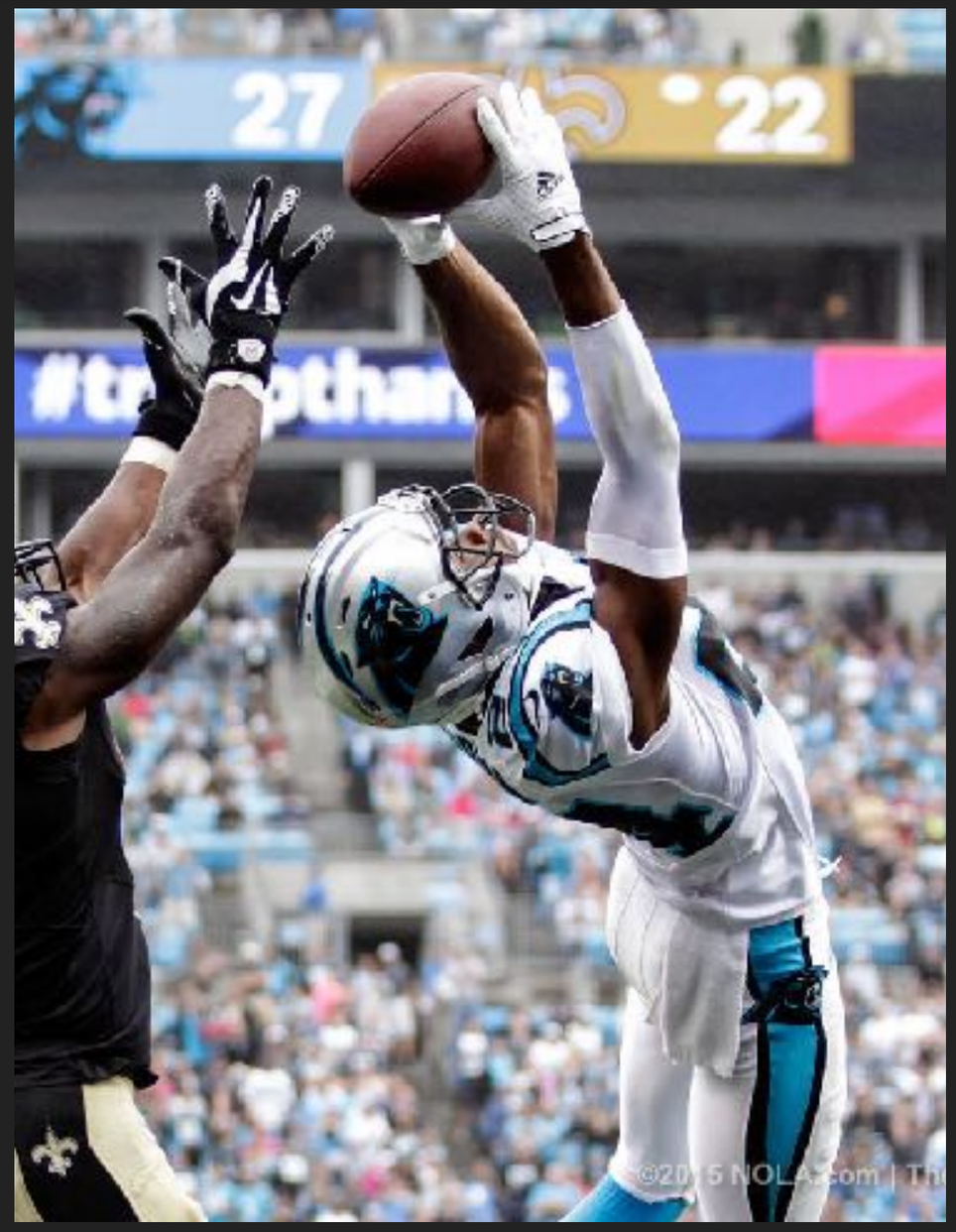
GITHUB REPO

- https://github.com/cnice/interceptor-app
- https://github.com/cnice/core.api



TOOLSET

- VS Code Editor
  - Goggle chrome debugger
- Visual Studio Community Edition
- ASP.NET Core API
  - JWT (Json Web Token)
- Postman

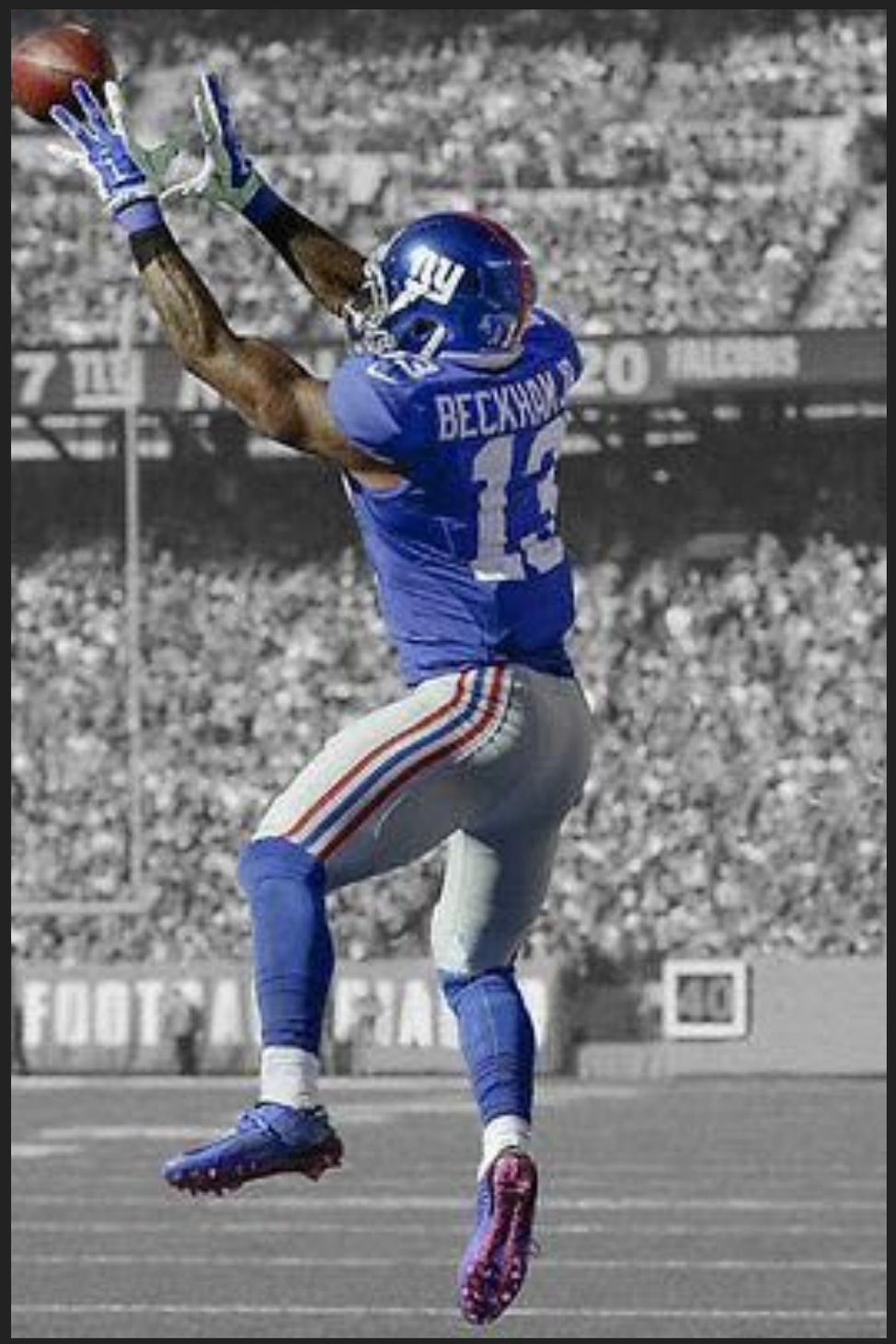


INTERCEPTOR FEATURE

- INTRODUCED WITH VERSION 4.3
- INTERCEPTORS SIT BETWEEN THE APPLICATION AND BACKEND (API).
- ALLOWS FOR INSPECTION AND TRANSFORMATION OF HTTP REQUESTS AND RESPONSES
  - INSPECT AND TRANSFORM TO THE SERVER
  - INSPECT AND TRANSFORM FROM THE SERVER BACK TO THE APPLICATION (WITH SAME INSPECTORS)
- INTERCEPTORS FORM A FORWARD-AND-BACKWARD CHAIN OF REQUEST/RESPONSE HANDLERS.
- ALLOWS DEVELOPERS TO WRITE A VARIETY OF TASKS FROM AUTHENTICATION TO LOGGING.
- WITHOUT INTERCEPTORS, THESE TASKS WOULD NEED TO BE WRITTEN FOR EACH HTTP CLIENT CALL.

HTTPCLIENT MODULE @ANGULAR/COMMON/HTTP

- Intercept Method
- Next Object
- HttpEvents
- Immutability



INTERCEPT METHOD

- Intercept Method
  - Transforms a request as an observable and afterwards returns the HTTP response.
  - Inspects requests on the way in and forwards the results to the handle() method of the *Next* object.

```
intercept(req: HttpRequest<any>, next: HttpHandler):
    Observable<HttpEvent<any>> {
    return next.handle(req);
  }
```

NEXT METHOD

- The *Next* object
  - In essence, represents the next interceptor in the chain of interceptors if any.
  - The final *next in the chain represents the backend handler responsible for calling the server.*
  - *Sends the requests, eventually receives the response.*

INTERCEPTOR ORDER

- Interceptor Order
  - Interceptors execute in the order they are provided.
  - Requests flow A => B => C , responses flow back C => B => A

```
/** Http interceptor providers */
export const httpInterceptorProviders = [
  { provide: HTTP_INTERCEPTORS, useClass: StatusCodesInterceptor, multi: true },
  { provide: HTTP_INTERCEPTORS, useClass: TokenInterceptor, multi: true },
  { provide: HTTP_INTERCEPTORS, useClass: LogInterceptor, multi: true}
];
```

IMMUTABILITY

- Immutability
  - HttpRequest and HttpResponse instance properties in a interceptor are readonly, rendering them immutable.
  - Immutability ensures interceptors see the same request for each try.
  - Typescript prevents from setting HttpRequest readonly properties. (clone first, then modify the request or response!)

```
// Typescript disallows the following assignment because req.url is readonly

req.url = req.url.replace('http://', 'https://');

// clone request and replace 'http://' with 'https://' at the same time

 const secureReq = req.clone({

 url: req.url.replace('http://', 'https://')

});
```

REFERENCES

- Obviously!
  - https://angular.io/guide/http#intercepting-requests-and-responses
  - https://medium.com/@ryanchenkie_40935/angular-authentication-using-the-http-client-and-http-interceptors-2f9d1540eb8
  - https://blog.angularindepth.com/insiders-guide-into-interceptors-and-httpclient-mechanics-in-angular-103fbdb397bf