**Frida script file load 방법 (1)**

프리다 CLI 환경을 통해 후킹을 진행하다보면 오타가 날 경우 긴 코드를 타이핑해야 하는 난감한 경우가 발생하곤 한다.

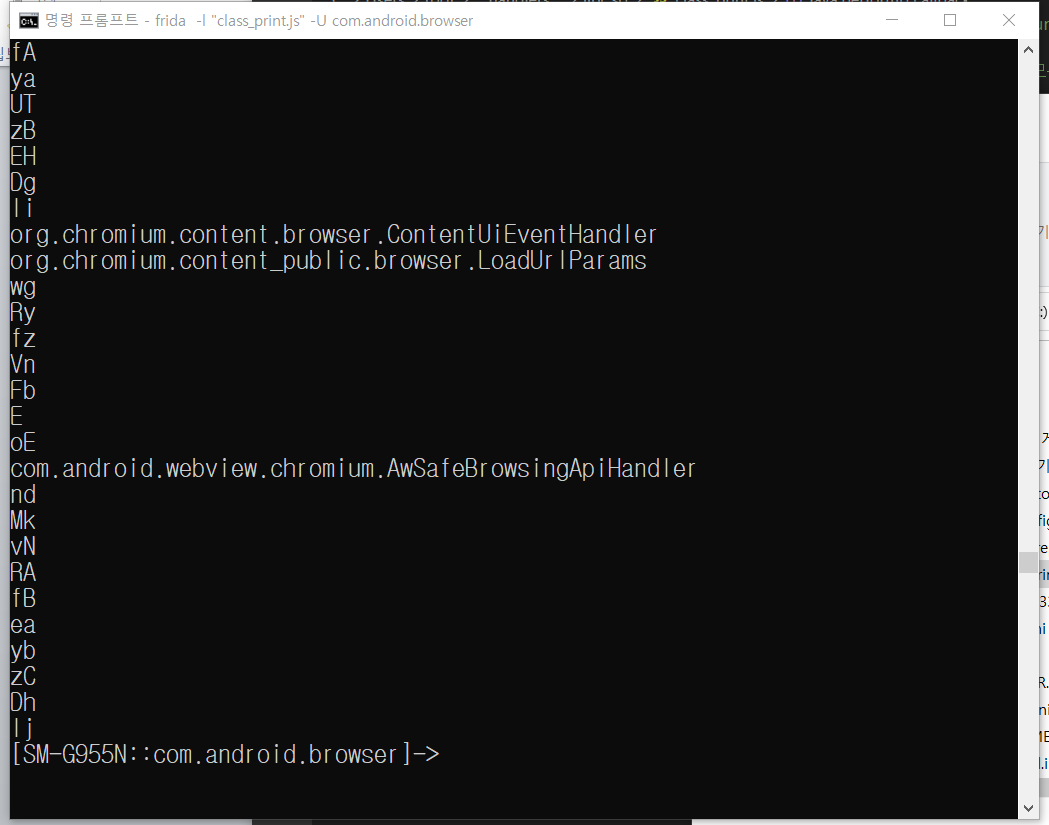
이에 따라, js 코드로 저장하여 사용이 가능하다.

js 스크립트 파일을 로드하여 후킹을 진행하면, 후킹 코드에 대한 유지보수가 용이해지고,

후킹할 스크립트가 메인 쓰레드가 실행되기 전에 삽입도 가능하게 된다.

1. 실행된 앱의 classname을 출력하는 간단한 js 후킹 코드를 작성해보자.

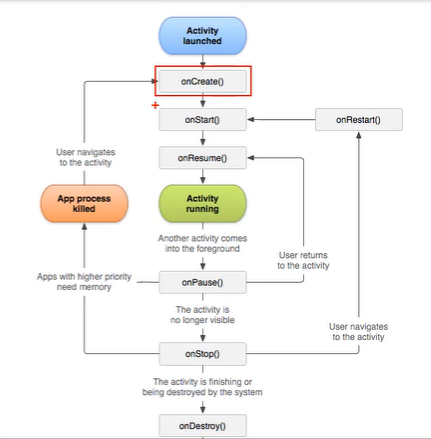
frida –l “class\_print.js” –U com.android.browser



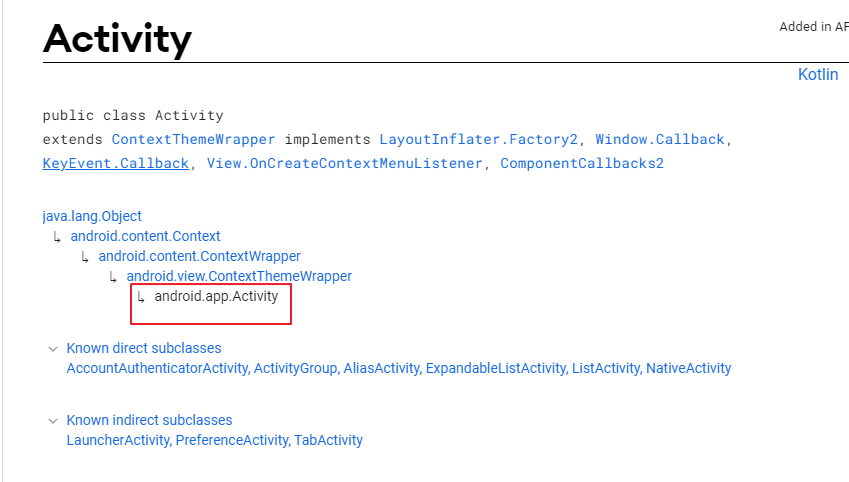
명령어를 실행하는 즉시 자바 스크립트 내 작성된 코드에 따라

해당 앱에 존재하는 classname을 CLI 화면에 출력해준다.

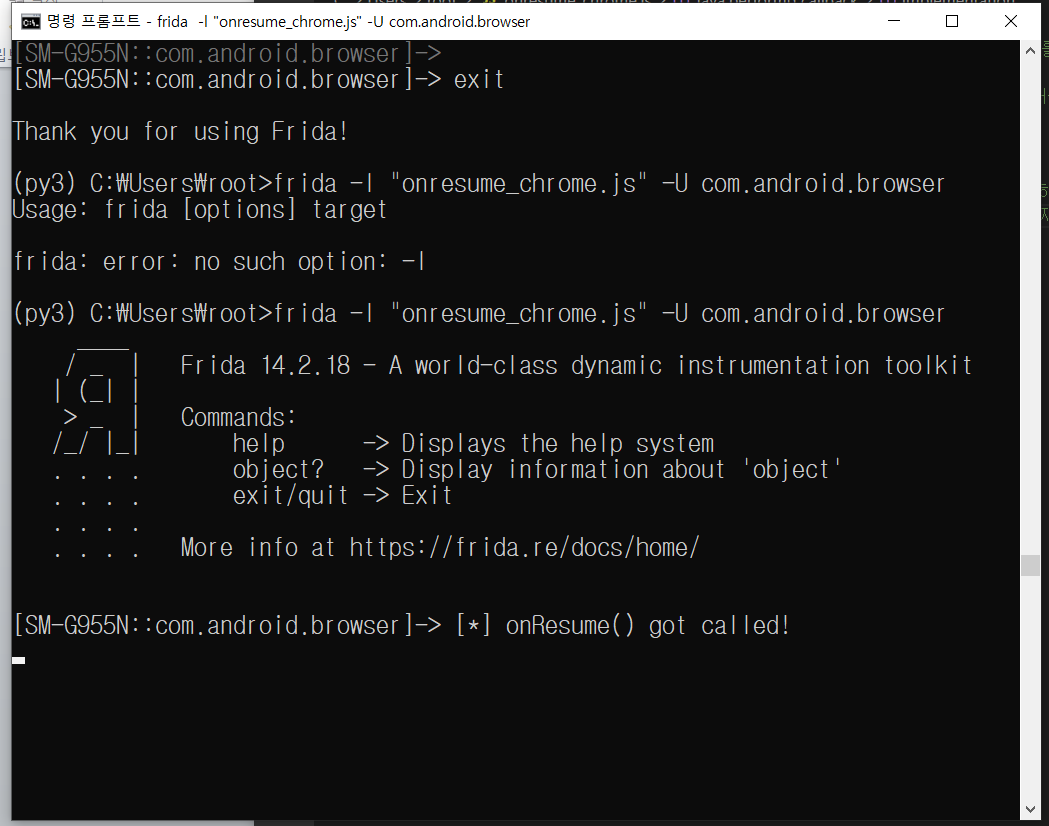
1. 안드로이드 생명주기를 보면 아래와 같다. onResume() 함수가 실행될 때마다 해당 여부를 출력해주는 js 코드를 만들어보자.



// onResume() 함수는 앱이 시작될 때도 발생하고, 홈 화면으로 이동 후 다시 접근할 경우에도 발생하게 된다.



구글에 검색하게 되면, 안드로이드 개발자 사이트를 통해 어떤 클래스를 사용하는지 확인이 가능하다.



이 또한, frida를 통해 browser 실행 후 해당 js 코드를 이용해 후킹해보자.

frida –l “onresume\_chrome.js” –U com.android.browser

백그라운드로 부터 포어그라운드로 앱이 이동하면 onResume() 함수의 호출 여부가 로그로 찍히게 된다.

Frida script file load 방법 (2)

이번에는 인스턴스화 된 객체를 Java.choose.API를 이용해 찾아보자.

힙에서 인스턴스화 된 android.view.View 인스턴스 객체를 찾아보자.

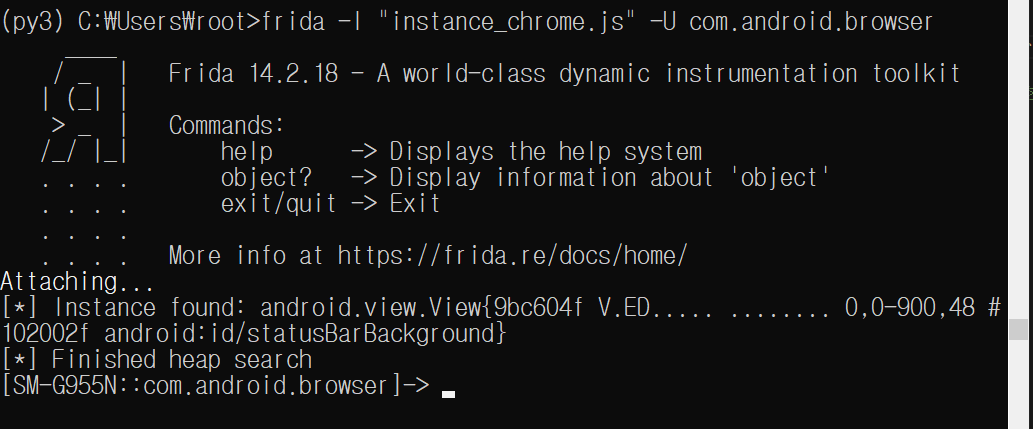
android.view.View

: 애플리케이션에서 눈에 보이는 이미지, 버튼 등을 가리킴



// js 코드가 정상적으로 후킹되어, 총 1개에 대한 인스턴스를 확인할 수 있었다.

(instance\_chrome.js) 코드를 약간 수정 시 instance에 대한 값을 텍스트로 확인할 수 있다.



마지막으로 정리하자면, Java.use()는 클래스 자체를 이용하는 개념이고,

Java.choose()는 이미 생성된 객체 인스턴스를 사용하는 개념이다.