

# Weinre을 사용한 Firefox OS 원격 디버깅 방법

다음 글

작성일: 2013/03/15 by Jaebeom

분류: Firefox Firefox OS 디버깅 모바일

생각 나누기

이 글은 Schalk Neethling과 Robert Nyman의 *Remote Debugging Firefox OS with Weinre*에 대한 번역 및 편집본입니다.

여러분이 Gaia에 기여하길 원하거나 Firefox OS의 웹앱을 만들길 원한다면 B2G 데스크톱버전이나 기기위에서 테스트할 때 직면하게 될 어려움은 HTML, CSS와 자바스크립트를 검사하고, 디버깅하기 위한 개발 도구의 부족일 겁니다.

현재 두가지 버그를 가지고 있고, 그들의 우선순위를 정하기 위해 투표를 진행하고 있고, 자체적인 원격 검사도구와 스타일 편집기의 개발을 추진하기 위해 몇가지 좋은 소식이 있습니다. 여러분은 원격 디버거에 접근할 수 있습니다.

그리고 어떻게 가능한지 알고 싶어 할 것이라고 생각합니다. 첫 번째 단계 :Weinre. Weinre는 Apache 재단의 프로젝트이고 웹 원격 검사기(WEb Inspector REmote)의 표준이고 정확히 이름이 의미하듯이 Firebug나 Webinspect와 같은 도구이지만 웹페이지를 원격에서 실행하고 디버깅할 수 있습니다. 그래서 여러분이 Firefox 개발자도구 혹은 Chrome 개발자도구를 사용해왔다면 Weinre는 쉽게 사용하실 수 있습니다. 말로만 얘기하기엔 충분하지 않으니 이걸 사용해 보기로 하겠습니다.

## Weinre 설정하기

Weinre는 Node.js 위에서 실행되기에 여러분은 Node.js를 설치하여야 합니다. Node.js는 NPM이라는 패키지 매니저를 가지고 있고, Weinre를 설치하기 위해 사용하게 될 것입니다. 터미널에 다음 구문을 실행하십시오:

```
npm -g install weinre
```

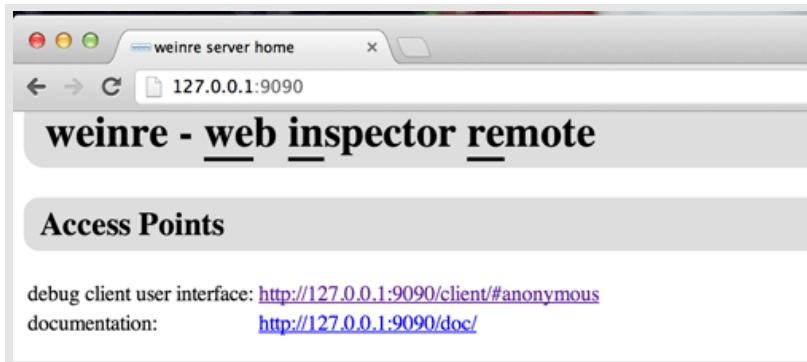
노트: -g 플래그는 Weinre를 모든 사용자가 사용할 수 있는 모듈로 설치되도록 설정합니다. Linux와 Mac에서는 sudo를 명령어 앞에 붙여서 실행할 필요가 있습니다.

설치과정이 끝나면 디버깅하기 위해 Weinre를 사용할 준비가 됩니다. 그러나 처음으로 Weinre가 완전히 성공적으로 설치 되었는지 확인해봅시다:

```
$ weinre --boundHost 127.0.0.1 --httpPort 9090  
2013-01-28T10:42:40.498Z weinre: starting server at http://127.0.0.1:9
```

여러분은 앞에 나온 내용과 비슷한 응답을 받았다면 성공적으로 설치되었고 Weinre 서버가 실행됩니다. 브라우저를 켜고 <http://127.0.0.1:9090>을 입력하세요. (노트: Weinre에 대한 UI는 Webkit를 위해 만들어져 있습니다. 그래서 이것은 다른 브라우저에서는 정상적으로 작동하기 전까지 여러분은 Chrome을 사용할 걸 권함

니다.)



위는 Weinre 클라이언트와 문서에 접근할 수 있는 Weinre 서버의 방문페이지입니다. 디버깅 클라이언트 링크를 클릭하세요.



여러분이 보셨다시피 한 개의 클라이언트가 연결되었고 웹검사기와 몇몇 일반적인 서버 속성을 나타냅니다만 target은 없습니다. 이제 target을 설정해봅시다.

노트: Winery의 UI가 매우 친근하게 느끼신다면 Winery가 Chrome과 Safari의 웹검사기와 동일한 UI를 사용하기 때문일 겁니다.

## Weinre target 설정하기

Weinre에서 target은 여러분이 디버깅하길 원하는 웹페이지나 앱이고, 접속 가능한 target을 만들기 위해 앱의 관계된 파일 안에 한 줄을 추가해야 합니다. 달력앱을 검사해보도록 하겠습니다. gaia>apps>calendar->index.html를 여시고 맨 아래줄로 이동하세요. body 태그가 닫히기 전에 다음 줄을 추가하세요:

```
<script src="http://127.0.0.1:9090/target/target-script-min.js#anonymo
```

B2G 데스크톱 버전을 올리고, 디버깅을 시도하기 전에 해야 할 한 가지가 더 있습니다. Gaia는 보안정책(Content Security Policy)을 사용하고 스크립트가 애플리케이션일 경우 동일한 영역에서만 로드할 수 있도록 제한하고 있습니다. 그래서 Calendar를 로드하고 디버깅하려면 특별한 영역에서 로딩을 못하도록 되어 있는 부분을 해제해야 합니다.

이 작업을 하기 위해 임시적으로 CSP(Content Security Policy)를 불가능하도록 해

야 합니다. gaia->build->preferences.js를 열고 약 24라인 쯤에 다음 줄을 추가하세요:

```
prefs.push(["security.csp.enable", false]);
```

## Weinre 과 B2G 데스크톱 버전을 사용해서 디버깅하기

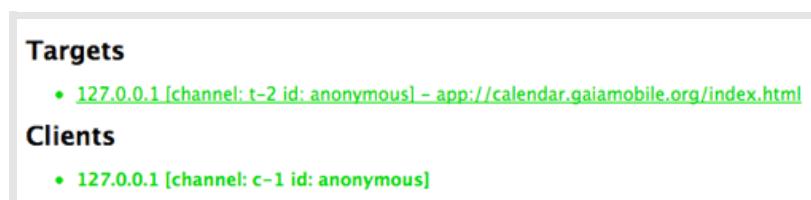
Weinre를 시험할 준비가 되었습니다. 여러분은 Gaia 최상위 디렉토리에 있지 않다면 최상위 디렉토리로로 변경하고 실행하세요:

```
DEBUG=1 make
```

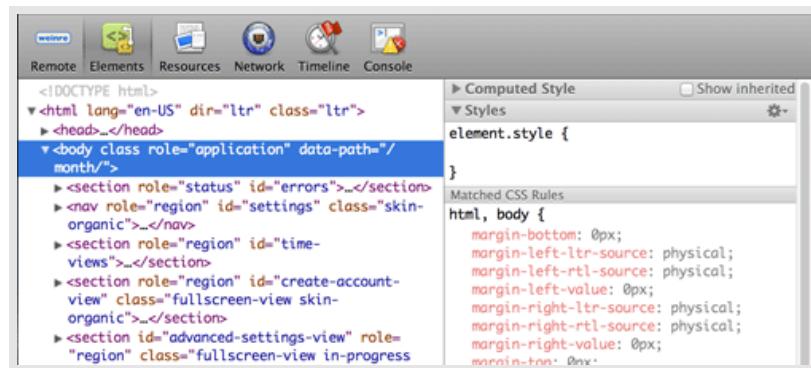
profile을 만들고 B2G 데스크톱버전을 실행하세요:

```
/Applications/B2G.app/Contents/MacOS/b2g-bin -profile /Users/username/
```

B2G가 실행되면 스크린을 열고 오른쪽으로 두번 스크린을 이동시키고 Calendar앱을 띄우세요. 앱이 실행되면 Weinre의 클라이언트 대시보드에 새로운 아이템이 나오게 됩니다:



여러분이 보시다시피 Calendar는 성공적으로 연결되고 target 중 하나로 나오게 됩니다. 'Elements' 탭을 클릭해보세요.



왼쪽에는 앱의 HTML코드가, 오른쪽에는 CSS가 나열됩니다! 여러분은 HTML, CSS에 직접 움직이거나 수정할 수 있고 변경사항이 바로 적용됩니다. 회색이고 사용하지 않는 요소라도 수정할 수 있습니다. 여러분은 또한 elemnet.style을 추가하거나 현재 방식을 수정하여 새로운 스타일을 추가할 수 있습니다. 여러분은 또한 개발한 요소뿐만 아니라 브라우저에서 생성한 스타일에도 접근할 수 있습니다.

## 콘솔로 작업하기

우리의 흥미를 이끈 다음 탭은 Console 탭입니다. 여기에서 여러분은 직접 접근하기 원하는 앱의 자바스크립트를 작성하고 실행해 볼 수 있습니다. 어떻게 이것이 가능한지 살펴보기 위해 Dialer의 로그의 일부를 확인해보겠습니다.

첫번째로 Calendar에서 Dialer로 스크립트를 이동시킵니다. Calendar에서 코드를 복사하고 gaia -> apps -> communication -> dialer -> index.html을 열고 코드를 붙여넣으세요. 다음으로 여러분의 프로파일을 ‘make’를 사용해서 다시 구축(build)하고 마지막으로 B2G 데스크톱버전을 새로 실행하세요.

다시 실행되면 홈스크린의 왼쪽 아래에 있는 Dialer를 누르십시오. 완전히 로드되면 <http://127.0.0.1:9090/client/#anonymous>를 통한 Weinre에게 통신 채널을 열 것을 승인하고 target이 승인되어 지는 상태를 다음과 같이 볼 수 있습니다:

```
127.0.0.1 [channel: t-7 id: anonymous] - app://communications.gaiamobi
```

dialer를 열고 왼쪽 아래부분에 있는 call log를 클릭하세요. 현재 call log에 대한 몇몇 견본 데이터가 유명하지만 우리 자신의 데이터를 만들어 봅시다. Weinre의 console 탭을 열고, 다음과 같이 입력해주세요.

```
RecentsDBManager.deleteAll();
```

여러분의 데스크톱에서는 아직 아무것도 보이지 않을 것이고, 더 진행할 부분이 있습니다. 다음과 같이 입력해주세요.

```
Recents.refresh();
```

아하! 여러분이 봤듯이 우리의 call log는 비었습니다. 다음 단계로 엔트리를 추가할 것입니다. 이것을 위해서 우리는 더미 콜 엔트리 오브젝트(dummy call entry object)를 만들고 이것을 저장하기 위한 RecentsDBManager의 함수를 추가할 것입니다:

```
// Dummy entry
var recentCall = {
  type: 'incoming-refused',
  number: '555-6677',
  date: new Date()
};
RecentsDBManager.add(recentCall);
Recents.refresh();
```

여러분이 보셨다시피, 생성된 엔트리는 **IndexedDB**에 추가되었고, call log에서 볼 수 있습니다. 여러분이 이미 발견했다시피 유용한 기술 중 하나는 개발 속도를 높일 수 있는 자동완성 기능입니다.

Firefox OS위에서 작동하고 앱을 작성하는 기능들의 조합은 프로파일을 작성하고 분해하고, B2G를 다시 올리는데 시간과 노력을 아껴줍니다.

그러나 잠깐, 기기위에서의 디버깅은 어떠할까요? 다른 아이피라는 차이점을 제외하면 위와 동일하게 작동할 것입니다. 여러분이 기기에서 디버깅하길 원한다면 여러분은 여러분의 호스트 컴퓨터의 IP 주소를 알아야 합니다. 그리고나서 여러분은 연결된 호스트로 이 IP를 사용하여 Weinre를 시작하고 대상 문서에 스크립트를 포함할 때 이 IP를 포함해야 합니다.

Linux와 Mac에서는 ifconfig를 사용하여 주소를 얻을 수 있고 Windows에서는 ipconfig를 사용할 수 있습니다. 새로운 IP를 가지면 Weinre의 현재 인스턴스를 정지하고 다음과 같이 입력해야 합니다:

```
weinre --boundHost 192.168.1.1 --httpPort 9090
```

그리고 대상 문서에는 다음을 포함해야 합니다:

```
<script src="http://192.168.1.1:9090/target/target-script-min.js#anonymous">
```

여러분의 프로파일을 만들고 기기에 밀어 넣으려면 다음과 같이 입력하세요.

```
make install-gaia
```

여러분의 앱을 구동시키고 여러분의 앱을 세상에 알리세요!

## 결론

이 방법이 완벽한 것은 아니어서 여러분은 소스 저장소에 보내기 전에 여러분의 변경사항을 되돌릴 지점을 기억해야 합니다. 매번 수동으로 스크립트를 추가해야 하고 완벽히 작동하지 않을 수 있습니다. 마우스를 올렸을 때 변경되는 highlighting DOM 요소와 breakpoint로 디버깅하려는 자바스크립처럼, 그리고 Gaia에서 직접 작업할 뿐만 아니라 Firefox OS를 위해 흥미로운 앱을 개발하려는 모든 개발자의 삶을 향상시키려면 오랜 기간이 걸릴 것입니다.

그러나 스크립트의 injection을 관리함에 대해 manually하게 벌써 약간의 빛이 보입니다. CSP를 불활성화하고 몇가지를 명확하게 함은 소스 저장소에 보내기전에 마무리해야 한다. Jan Jongboom은 매우 가능성있고 많이 편리해질 수 있도록 해주는 소스를 Gaia 소스 저장소에 밀어넣기(pull) 요청을 했습니다. 이러한 작업은 계속될 것이고 그에게 힘을 주어 Gaia안에 포함되도록 도와주십시오. 행복한 개발시간을!

중요 내용 : Kevin Grandon이 Weinre의 사용법을 기억하고 이메일을 보내지 않았다면 위와 같은 일은 일어나지 않았을 겁니다. Kevin 고마워요!

