Hanbit eBook

Realtime 09

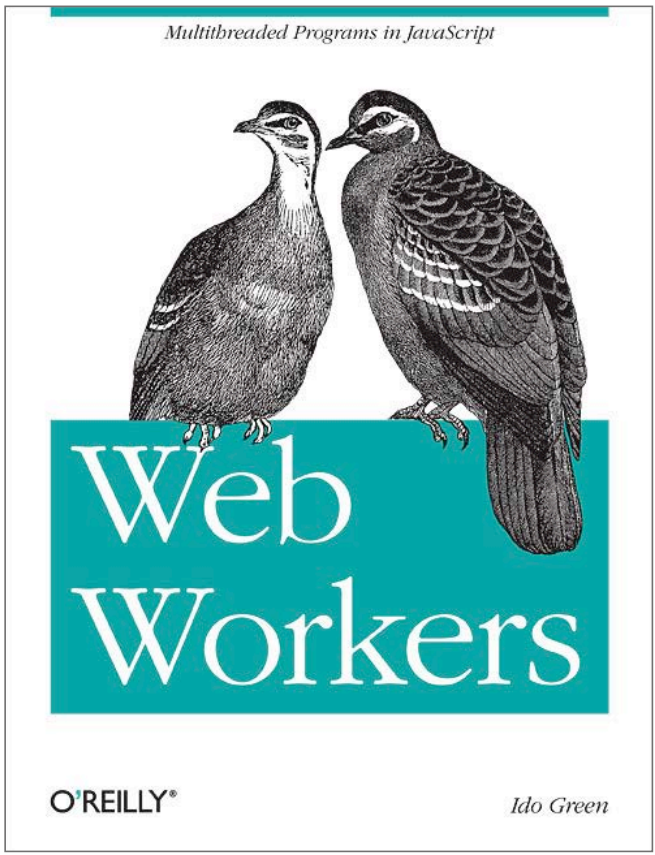
멀티스레드를 위한 자바스크립트 프로그래밍

웹 워커

Web Workers

아이두 그린 지음 / 김보경 옮김

O'REILLY® 한빛미디어 Hanbit Media, Inc.



이 도서는 O'REILLY의
Web Workers의
번역서입니다.

멀티스레드를 위한 자바스크립트 프로그래밍

웹 워커

멀티스레드를 위한 자바스크립트 프로그래밍 **웹 워커**

초판발행 2012년 11월 23일

지은이 아이두 그린 / **옮긴이** 김보경 / **펴낸이** 김태헌
펴낸곳 한빛미디어(주) / **주소** 서울시 마포구 양화로 7길 83 한빛미디어(주) IT출판부
전화 02-325-5544 / **팩스** 02-336-7124
등록 1999년 6월 24일 제10-1779호
ISBN 978-89-7914-978-4 15560 / **정가** 8,900원

책임편집 배용석 / **기획** 김창수 / **편집** 김병희
디자인 표지 여동일, 내지 스튜디오 [밈], 조판 김현미
영업 김형진, 김진불, 조유미 / **마케팅** 박상용, 박주훈, 정민하

이 책에 대한 의견이나 오탈자 및 잘못된 내용에 대한 수정 정보는 한빛미디어(주)의 홈페이지나 아래 이메일로 알려주십시오.
한빛미디어 홈페이지 www.hanb.co.kr / **이메일** ask@hanb.co.kr

지금 하지 않으면 할 수 없는 일이 있습니다.
책으로 펴내고 싶은 아이디어나 원고를 메일(ebookwriter@hanb.co.kr)로 보내주세요.
한빛미디어(주)는 여러분의 소중한 경험과 지식을 기다리고 있습니다.

지은이_ **아이두 그린**

아이두 그린은 구글 크롬 OS의 '디벨로퍼 애드보컷Developer Advocate'*이다. 15년 이상의 개발 경력을 가진 그는 웹 애플리케이션 개발을 좋아하는데, 특히 amazing UX를 이용하여 웹 애플리케이션을 개발하는 것을 좋아한다. 자바·php·펄·자바스크립트 등 다양한 언어에 대한 기술과 경력을 보유하고 있는 동시에 애자일 개발 능력 및 시스템 스케일링 전반에 대한 지식의 소유자이기도 하다.

옮긴이_ **김보경**

숭실대학교에서 정보통신공학을 전공했다. 졸업 후에는 삼성SDS의 R&D 부서에서 근무하다 쇼핑몰을 운영하기도 했다. 그 후 공학에서 어문학으로 진로를 180도 변경, 이화여대 통역번역대학원 한영번역과에 입학하면서 번역가의 길로 들어섰다. 다양한 기관의 한영 및 영한 기술번역을 주로 맡아 왔으며 현재는 월스트리트 저널 코리아에서 영한번역가로 근무 중이다.

---

* Developer Advocate은 외부에 구글Google의 신기술을 홍보하고 구글 내에서는 개발자의 요구를 대변합니다. 그리고 Chrome, Android, App Engine, Wave, Map API, HTML5, Apps 및 Ads API와 같은 제품이 더욱 혁신적인 성과를 이뤄낼 수 있도록 지원합니다. 자세한 내용은 "http://googlekorcam.blogspot.kr/2010/10/developer-advocate-seoul.html"를 참조하세요.

저자 서문

웹 워커는 매우 강력한 HTML5 기능임에도 불구하고 그동안 많은 관심을 받지 못했다. 웹 워커에서 제공하는 API를 사용하면 자바스크립트를 별도의 스레드에서 실행할 수 있어, 웹 애플리케이션의 사용자 인터페이스UI(User Interface)에 영향을 주지 않는다. 이 경우 자바스크립트는 메인 렌더러는 물론, 그 위에서 구동되는 모든 UI와 동시에 실행된다. 따라서 리소스를 많이 요구하는 긴 작업을 실행하는 동안 페이지를 무반응 상태로 두는 일이 없게 된다.

다른 기술에서 사용되는 스레드와 마찬가지로 웹 워커도 상대적으로 무거운 편이다. 각각의 워커가 상당한 시스템 리소스를 차지하므로 한 번에 많이 사용하지 않는 것이 좋다. 웹 워커는 CPU·네트워크·대역폭 등 제한된 리소스에 의존하는 긴 작업을 처리하는 데 주로 사용되며, 초기 비용 및 메모리 사용량 역시 높은 편이다.

웹 워커는 진화하고 있는 새로운 표준인 만큼, 웹 브라우저마다 웹 워커 스펙을 구현하는 방식에 차이가 있다. 일부 기능은 안정 단계에 접어들었다고는 하나, 가까운 미래에 IndexedDB 접근과 같은 기능이 대부분의 모던 웹 브라우저에서 제공될지는 의문이다. 필자는 이 책의 보급 및 모던 웹 브라우저의 도입을 통해 이처럼 막강한 API의 사용이 더욱 확대될 수 있기를 기대한다.

저자 **아이두 그린**

역자 서문

IT 회사에서 근무해 본 경험이 있는 역자는 IT 환경이 얼마나 급변하는지 잘 알고 있다. 자고 일어나면 새로운 제품, 새로운 기술이 쏟아져 나온다고 해도 과언이 아닐 것이다. 특히 최근 몇 년 동안에는 스마트폰의 보급이 확대되면서 웹 플랫폼이 모바일 플랫폼으로 이동하고 이와 함께 앱시장도 급속도로 성장하고 있다. 이처럼 일변하는 환경 속에서 사용자들의 니즈에 부합하는 개발환경을 제공하는 신기술이 속속 등장하고 있다.

'웹 워커'는 속도와 안정성 면에서 탁월한 성능을 제공하는 획기적인 HTML5 기술이다. 무거운 연산들을 UI와 완전히 분리시켜 백그라운드 단에서 수행함으로써 사용자의 대기시간을 획기적으로 단축시켜 준다. 어렵고 복잡하다고 알려진 멀티스레딩 기법을 간단한 API를 통해 구현할 수 있다는 것도 웹 워커의 큰 장점이다.

앱 사용자들의 요구수준이 갈수록 높아지고 있는 만큼 프로그램의 막강한 성능은 이제 선택이 아닌 필수가 됐다. 그런 점에서 이 책은 개발자라면 반드시 읽어야 할 필독서라 할 수 있다. 이 책은 이해하기 어렵고 딱딱한 일반 기술서들과는 달리 웹 워커라는 최신 기술을 상당히 평이하게 설명하며 다양한 예제를 통해서 독자들의 이해를 돕고 있다. IT 업계에서 오랜 시간 떠나있던 역자가 이 책을 끝까지 번역해 낼 수 있었던 것도 바로 이 때문이 아니었을까 싶다.

번역 작업에 들어가기 전 역자는 학부 시절 전공서적들을 보면서 난감해했던 기억들을 떠올렸다. 비영어권 독자들에게 영어권 기술을 전수하고 이해를 돕기 위한 목적으로 만들어진 번역서가 오히려 원서보다도 이해하기 힘든 경우가 많았기 때문이다. 그러한 경험을 떠올리며 최대한 한국 독자들의 관점에서 번역하려 노력했다. 역자는 쓸데없이 어려운 말들을 남발해 읽는 이로 하여금 기죽게 만드는 '이해불가

성’ 글들을 좋아하지 않는다. 이에 최대한 명확하고 간결하게 작업하려 했던 역자의 의도가 효과적으로 전달되었으면 한다. 작업 기간 동안 많은 부분을 배려해 주신 한빛미디어에 감사드리며, 이 책을 읽는 독자 한 분 한 분의 건승을 기원한다.

번역을 마치며

김보경

대상 독자 및 도서 구성

초급 **초중급** 중급 중고급 고급

이 책에서 사용되는 툴들은 중급 이상의 자바스크립트 사용자를 대상으로 한다. 특히 이벤트 핸들링과 콜백에 대한 이해는 필수다.

웹 워커의 세계로 들어가기 전에 웹 워커의 정의와 기능에 대해 먼저 살펴본다. 그 다음, 여러분이 사용하는 웹 브라우저에서 이 기능을 지원하는지를 확인하는 방법을 'Web Worker Hello World' 예제와 함께 알아본다. 그 후의 장에서는 다양한 종류의 웹 워커(전용dedicated, 공유shared, 인라인inline 등)를 다루면서 각 웹 워커를 사용하는 방법 및 특정 작업에 가장 적합한 웹 워커를 선택하는 방법을 배운다. 그런 다음 대표적인 웹 워커 디버깅 사례들을 소개한다. 마지막으로, 웹 워커가 웹 브라우저 외부에서 사용되는 기능임을 감안해 서버측 노드 환경에 적용하는 방법을 살펴본다.

예제 파일

예제 파일은 “https://github.com/greenido/Web-Workers-Examples-”에서 받을 수 있다.

이 책에서 사용되는 모든 예제는 크롬(15+)와 파이어폭스(7+)에서 테스트하였다. 또한 웹 워커는 모바일 사파리(5+) 에서도 지원되며, 아이폰 4GS부터 멀티코어 프로세서를 탑재한 만큼 더욱 유용한 기능이 될 것으로 보인다.

한빛 eBook 리얼타임

한빛 eBook 리얼타임은 IT 개발자를 위한 eBook 입니다.

요즘 IT 업계에는 하루가 멀다 하고 수많은 기술이 나타나고 사라져 갑니다. 인터넷을 아무리 뒤져도 조금이나마 정리된 정보를 찾는 것도 쉽지 않습니다. 또한 잘 정리되어 책으로 나오기까지는 오랜 시간이 걸립니다. 어떻게 하면 조금이라도 더 유용한 정보를 빠르게 얻을 수 있을까요? 어떻게 하면 남보다 조금 더 빨리 경험하고 습득한 지식을 공유하고 발전시켜 나갈 수 있을까요? 세상에는 수많은 종이책이 있습니다. 그리고 그 종이책을 그대로 옮긴 전자책도 많습니다. 전자책에는 전자책에 적합한 콘텐츠와 전자책의 특성을 살린 형식이 있다고 생각합니다.

한빛이 지금 생각하고 추구하는, 개발자를 위한 리얼타임 전자책은 이렇습니다.

1. **eBook Only - 빠르게 변화하는 IT 기술에 대해 핵심적인 정보를 신속하게 제공합니다.**

   500페이지 가까운 분량의 잘 정리된 도서(종이책)가 아니라, 핵심적인 내용을 빠르게 전달하기 위해 조금은 거칠지만 100페이지 내외의 전자책 전용으로 개발한 서비스입니다. 독자에게는 새로운 정보를 빨리 얻을 수 있는 기회가 되고, 자신이 먼저 경험한 지식과 정보를 책으로 펴내고 싶지만 너무 바빠서 엄두를 못 내시는 선배, 전문가, 고수분에게는 보다 쉽게 집필하실 기회가 되리라 생각합니다. 또한 새로운 정보와 지식을 빠르게 전달하기 위해 O'Reilly의 전자책 번역 서비스도 준비 중이며, 조만간 선보일 예정입니다.

2. **무료로 업데이트되는, 전자책 전용 서비스입니다.**

   종이책으로는 기술의 변화 속도를 따라잡기가 쉽지 않습니다. 책이 일정한 분량 이상으로 집필되고 정리되어 나오는 동안 기술은 이미 변해 있습니다. 전자책으로 출간된 이후에도 버전 업을 통해 중요한 기술적 변화가 있거나, 저자(역자)와 독자가 소통하면서 보완되고 발전된 노하우가 정리되면 구매하신 분께 무료로 업데이트해 드립니다.

3. 독자의 편의를 위하여, DRM-Free로 제공합니다.

구매한 전자책을 다양한 IT기기에서 자유롭게 활용하실 수 있도록 DRM-Free PDF 포맷으로 제공합니다. 이는 독자 여러분과 한빛이 생각하고 추구하는 전자책을 만들어 나가기 위해, 독자 여러분이 언제 어디서 어떤 기기를 사용하시더라도 편리하게 전자책을 보실 수 있도록 하기 위함입니다.

4. 전자책 환경을 고려한 최적의 형태와 디자인에 담고자 노력했습니다.

종이책을 그대로 옮겨 놓아 가독성이 떨어지고 읽기 힘든 전자책이 아니라, 전자책의 환경에 가능한 최적화하여 쾌적한 경험을 드리고자 합니다. 링크 등의 기능을 적극적으로 이용할 수 있음은 물론이고 글자 크기나 행간, 여백 등을 전자책에 가장 최적화된 형태로 새롭게 디자인하였습니다.

앞으로도 독자 여러분의 충고에 귀 기울이며 지속해서 발전시켜 나가도록 하겠습니다.

차례

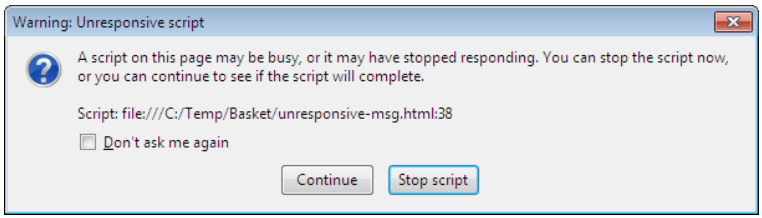
1 | 웹 워커의 개요

무거운 계산을 백그라운드에서 실행할 수 있다면 계산이 끝날 때까지 사용자 인터페이스UI(User Interface)가 기다릴 필요가 없게 되고, 많은 경우 모던 웹 애플리케이션의 성능이 향상될 것이다. 이처럼 무거운 계산을 수행하는 코드를 웹 애플리케이션의 UI와 별도의 스레드에서 실행시키는 API가 웹 워커 스펙[01]에 정의되어 있다. 웹 워커는 별도의 스레드로 실행되기 때문에 UI가 사용하는 메모리와 CPU에 영향을 주지 않고 긴 작업을 수행할 수 있다.

멀티스레드 프로그래밍은 복잡한 알고리즘과 이론적인 논의 등을 수반하는 까다로운 작업이라고 할 수 있다. 자바를 비롯한 여러 언어가 이러한 복잡함[02]을 숨기기 위한 라이브러리를 개발자들에게 제공한다. 다행히도 웹 워커에서는 훌륭하면서도 간단한 API를 제공하는데, 이 API를 사용하면 교착상태deadlock[03]나 그와 유사한 문제에 대해 신경을 쓰는 일이 줄어들어 그만큼 효율성이 증대된다.

웹 워커는 그림 1-1과 그림 1-2에서 보는 것처럼 '스크립트 반응없음' 대화상자가 더 이상 뜨지 않도록 해 줄 것이다.

그림 1-1 윈도우의 '스크립트 반응없음' 경고 다이얼로그



01 http://www.whatwg.org/specs/web-apps/current-work/multipage/workers.html

02 http://gee.cs.oswego.edu/dl/concurrency-interest/index.html - 멀티스레딩에 관해 더 많은 정보를 얻고자 한다면 "http://en.wikipedia.org/wiki/Multithreading_(computer_architecture)"을 방문해 볼 것.

03 (역자주) 교착상태('데드락'이라고도 함)이란 두 개 이상의 작업이 서로 상대방의 작업이 끝나기 만을 기다리고 있기 때문에 결과적으로 아무것도 완료되지 못하는 상태를 가리킨다. 전산학에서 교착상태란 다중 프로그래밍 환경에서 흔히 발생할 수 있는 문제로, 이 문제를 해결하는 일반적인 방법은 아직 없다.

그림 1-2 맥 OS의 '스크립트 반응없음' 경고 다이얼로그

1.1 웹 워커로 할 수 있는 작업

웹 애플리케이션에서 150밀리세컨드(0.15초) 이상 걸리는 긴 작업을 완료해야 하는 경우라면 웹 워커의 사용을 고려해 보아야 할 것이다. 이때 웹 애플리케이션이 네이티브 애플리케이션처럼 느껴지게 하기 위해서는 소요 시간을 80밀리세컨드 내외로 맞춰야 한다. 물론 이 시간은 웹 브라우저와 하드웨어에 따라 좌우된다. 특히 모바일 웹 애플리케이션을 개발할 때는 소요 시간을 더 줄여야 하는데, 모바일 기기의 CPU는 데스크톱에 비해 성능이 떨어지기 때문이다.[04] 소요 시간을 조정하기 위한 작업에 앞서, 여러분의 웹 애플리케이션에서 다음 작업들을 수행하는지 살펴보는 것이 좋다.

- 긴 문자열의 인코딩/디코딩
- 복잡한 수학 계산(소수 검색, 암호화, 모의 어닐링simulated annealing[05] 등)
- 긴 배열의 정렬
- 네트워크 요청 및 결과 데이터 처리
- 로컬 스토리지에서 계산 및 데이터 조작

04 모바일 브라우저에서의 웹 워커 - "http://greenido.wordpress.com/2012/02/07/google-chrome-for-android-is-out-there/"

05 (역자주) 모의 어닐링이란 커다란 탐색공간에서 주어진 함수의 전역 최적점에 대한 근사치를 찾는 전역최적화 문제에 대한 일반적인 확률적 휴리스틱probabilistic heuristic 접근방식이다.

- 프리페칭prefetching[06] 및 데이터 캐싱
- 문법 강조 또는 기타 실시간 텍스트 분석(철자 체크 등)
- 이미지 처리
- 비디오나 오디오 데이터의 처리 또는 분석(안면 및 음성 인식 포함)
- 백그라운드 I/O
- 웹 서비스 폴링polling[07]
- 긴 배열 또는 대용량 JSON 응답 처리

1.2 워커[08] 생성하기

웹 워커 API를 사용해 새로운 워커를 생성하려면 해당 스크립트를 불러오기만 하면 된다. 예를 들면 다음과 같다.

```
var worker = new Worker("worker.js");
```

위의 코드는 'worker.js'라는 스크립트를 로딩하여 백그라운드에서 실행한다. 워커 스레드는 실행할 스크립트의 URI를 매개변수로 하여 Worker()라는 생성자를 호출한다. 워커에서 데이터(처리된 정보의 출력, 통보 등)를 받을 때에는 워커의 onmessage 속성을 이벤트 핸들러 함수로 설정해야 한다. 다음 예제를 살펴보자.

```
var worker = new Worker('routes.js');
worker.onmessage = function(event) {
   console.log("Called back by the routes-worker with the best route to the pub");
}
```

06 (역자주) 프리페칭이란 웹 페이지에 있는 IP 주소를 미리 조회하여, 웹 페이지의 있는 링크를 더 빠르게 로드하는 방법이다.

07 컴퓨터 또는 단말 제어 장치 등에서 여러 개의 단말 장치에 대하여 순차적으로 송신 요구의 유무를 문의하고, 요구가 있을 경우에는 그 단말 장치에 송신을 시작하도록 명령하며 없을 때에는 다음의 단말 장치에 대하여 문의하게 되는 전송 제어 방식이다. _출처: TTA(http://word.tta.or.kr)

08 (역자주) 웹 워커가 생성하는 스레드를 '워커'라고 한다.

addEventListener를 사용해 워커와 계속 통신할 수도 있다.

```
var worker = new Worker('routes.js');
worker.addEventListener('message', function(event) {
  console.log("Called back by the routes-worker... with the best route to
    the pub")
}, false);
worker.postMessage(); // 워커를 시작한다.
```

1.3 웹 워커로 할 수 있는 것과 할 수 없는 것

워커는 부모 페이지의 DOM에 접근할 수 없다. 또한 다음 항목에 대해서도 접근 권한이 없다.

- window 객체
- document 객체
- parent 객체
- jQuery처럼, 워커도 실행 할 때 위의 객체들을 필요로 하는 자바스크립트 라이브러리들을 사용할 수 없다.

멀티스레딩이라는 속성 상, 웹 워커는 사용할 수 있는 자바스크립트 기능이 한정적이다. 사용 가능한 기능은 다음과 같다.

- navigator 객체
- location 객체 (읽기 전용)
- XMLHttpRequest 함수
- Base 64 ASCII와 2진 데이터 간의 상호 변환을 위한 atob() 및 btoa() 함수
- setTimeout() / clearTimeout() 및 setInterval() / clearInterval()

- dump()
- 애플리케이션 캐시
- importScripts() 메서드를 사용하는 외부 스크립트
- 기타 웹 워커 생성[09]

1.4 워커 실행

웹 워커 스레드들은 각 코드를 처음부터 끝까지 동기적으로 실행한 뒤, 이벤트와 타이머에 반응하는 비동기 단계로 진입한다. 바로 이 때문에 웹 워커는 크게 두 가지 종류로 나뉜다.

- onmessage 이벤트 핸들러를 등록하는 웹 워커: 백그라운드에서 실행해야 하는 긴 작업에 적합하다. 새로운 메시지를 계속 모니터링해야 하므로 이러한 웹 워커는 종료되지 않는다.
- onmessage 이벤트로 등록하지 않는 웹 워커: 메인 웹 애플리케이션 스레드에서 오프셋되어야 하는 단일 작업, 예를 들어 큰 JSON 객체를 페칭fetching하거나 파싱parsing[10]하는 작업 등에 사용된다. 이런 종류의 웹 워커는 해당 작업이 완료되는 즉시 종료된다(콜백이 등록된 경우, 웹 워커는 콜백이 완료될 때까지 기다린다).

1.5 웹 워커 API를 지원하는 웹 브라우저

표 1-1은 대부분의 모던 웹 브라우저에서 기본 웹 워커를 지원한다는 것을 보여준다. 심지어 모바일 웹 브라우저에서도 지원된다. 하지만 표 1-2를 보면 공유 웹 워커shared Web Workers를 지원하는 웹 브라우저는 많지 않다는 것을 알 수 있다. 이에 대해서는 5장에서 자세히 살펴본다.

---

09 http://www.html5rocks.com/en/tutorials/workers/basics/#toc-enviornment-subworkers

10 (역자주) 컴파일러나 인터프리터 같은 언어 해석기에서 입력으로 전달된 프로그램의 구조와 작업 내용을 이해하고 이를 기계어로 번역하기 위하여 프로그램의 문법에 정의된 내용에 따라 각 구성 요소의 구조를 알아내는 작업을 말한다.

표 1-1 웹 워커를 지원하는 웹 브라우저

웹 브라우저	버전
IE	10.0
크롬	12+
파이어폭스	5+
사파리	4+
오페라	11+
iOS 사파리	5+
오페라 모바일	11+
안드로이드	2.1
안드로이드용 크롬	Beta

표 1-2 공유 웹 워커를 지원하는 웹 브라우저

웹 브라우저	버전
IE	10.0은 지원여부 불확실
크롬	12+
파이어폭스	9+ 지원여부 불확실
사파리	5+
오페라	11+
iOS 사파리	5+
오페라 모바일	11+

2 | 웹 워커의 용도와 사용법

일부 웹 브라우저에서 지원되지 않는 기능을 사용할 경우에는 반드시 웹 브라우저 지원 여부를 확인해야 한다. 이와 마찬가지로 웹 워커를 사용하기 전에도 전역 윈도우 객체에 Worker 속성이 존재하는지를 확인해 보아야 한다. 웹 워커 API를 지원하지 않는 웹 브라우저에서는 Worker 속성을 정의할 수 없으므로 다른 방법을 찾아야 할 것이다(다행히 단계적 향상progressive enhancement 모델[01]을 기반으로 설계된 웹 애플리케이션에서는 그리 큰 문제는 아니다). 다음과 같은 간단한 헬퍼 함수helper function를 사용하면 웹 워커 지원 여부를 손쉽게 판단할 수 있다.

```
isWorkersAvailable() {
  return !!window.Worker;
}
```

위의 함수 대신 모더나이저 라이브러리(http://modernizr.com)를 사용해 클라이언트의 웹 브라우저에서 웹 워커를 지원하는지 판단하는 방법도 있다. 이 라이브러리는 웹 워커 지원 여부를 확인하여 지원하지 않는 경우 다른 방법을 모색하도록 해 준다.

```
if (Modernizr.webworkers) {
  // window.Worker 사용 가능!
} else {
  // 웹 워커 지원하지 않음
}
```

01 (역자주) 웹 페이지가 모든 웹 브라우저에서 똑같이 보여지도록 성능이 가장 낮은 웹 브라우저를 기준으로 개발하지 않고, 사용자에게 제공해야 할 필수 정보는 모두 제공하되 최신 웹 브라우저를 쓰는 사용자에게 더욱 좋은 화면 효과와 추가 기능을 제공한다는 개념이다.

지금까지 웹 브라우저의 웹 워커 지원 여부를 확인하는 방법을 알아보았다. 이제 웹 워커의 세계에서 자주 등장하는 간단한 소수 계산 프로그램을 예제 2-1과 예제 2-2에서 살펴보자. 그림 2-1은 실행 결과다.

예제 2-1 highPrime.js (웹 워커로 사용 가능한 무작위 대입brute-force 기법 소수 계산기)

```
//
// 소수를 찾는 간단한 방법
//
var n = 1;
search: while (true) {
  n += 1;
  for (var i = 2; i <= Math.sqrt(n); i += 1) {
    if (n % i == 0) {
      continue search;
    }
  }
  // 소수 발견!
  postMessage(n);
}
```

예제 2-2 highPrime.js (웹 워커를 위한 HTML 호스트)

```
<!DOCTYPE HTML>
<html>
  <head>
    <title>Web Worker: The highest prime number</title>
  </head>

  <style>
    #result {
```

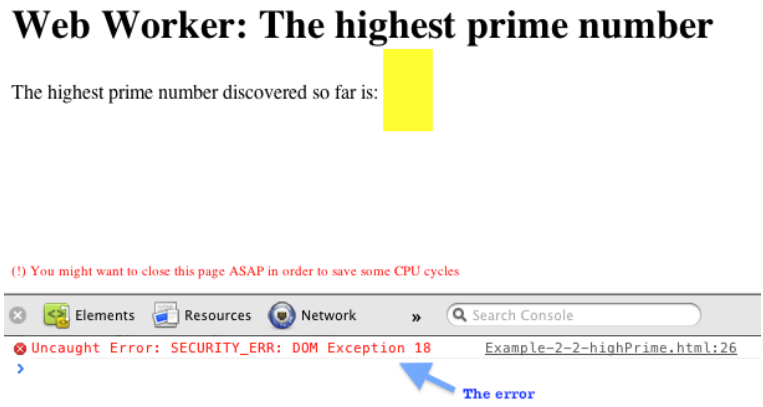
```
        background-color: yellow;
        padding: 20px;
        font-size: 140%;
      }
      footer {
        font-size: 70%;
        color: red;
        position: fixed;
        bottom: 1em;
        text-align: center;
      }
    </style>

    <body>
      <h1>Web Worker: The highest prime number</h1>
      <article>The highest prime number discovered so far is:
        <output id="result"></output>
      </article>
      <script>
        var worker = new Worker('highPrime.js');
        worker.onmessage = function (event) {
          document.getElementById('result').textContent = event.data;
        };
      </script>
    </body>
  </html>
```

NOTE_ 보안 규정으로 인해 크롬(16+)의 경우 로컬 환경(file:// 등)에서 웹 워커가 동작하지 않는다. 로컬 파일 및 file://을 사용하는 웹 애플리케이션을 실행할 경우에는 크롬을 --allow-file-access-from-files 플래그 세트와 함께 실행해야 한다. 일반 웹 브라우저를 이 플래그 세트와 함께 실행하는 것은 추천하지 않는다. 테스트나 개발 용도로만 사용하라.

NOTE_ 크롬17 이후 버전에서는 "Uncaught Error: SECURITY_ERR: DOM Exception 18"라는 메시지를 볼 수 있다. 이는 개발 시에는 로컬 서버에서 실행하라는 뜻이다. 그림 2-1에서 자세히 살펴보자.

그림 2-1 소수 생성 웹 워커에 대한 리포팅



postMessage()를 사용해 웹 페이지와 웹 워커 간에 전송된 메시지는 공유되는 것이 아니라 복사된다. 메인 웹 애플리케이션 페이지와 웹 워커는 동일한 객체 인스턴스를 가리키는 것이 아니라 메모리 사용기록의 복사본을 각각 보유한다. 모던 웹 브라우저에서 이것이 가능한 이유는 웹 애플리케이션 페이지와 웹 워커 각 단에 객체값을 인코딩/디코딩하는 JSON이 있기 때문이다. 다시 말해, JSON 혹은 기타 직렬화된 데이터를 전달할 수 있다는 뜻이다. 다음은 그 예다.

```
postMessage ({'cmd': 'start', 'time': Date.now() });
```