13

오디오 비디오 제어 및 위치 정보 서비스, 웹 워커

강의 목표

1. 자바스크립트와 <audio> 태그와 <video> 태그를 이용하여 오디오와 비디오 재생, 중단, 새로운 미디어 재생 등 다양한 제어를 할 수 있다.
2. HTML5에서 제공하는 위치 정보 서비스를 이해한다.
3. 자바스크립트 코드로 PC나 모바일 장치의 현재 위도 경도 위치를 알아낼 수 있다.
4. 자바스크립트 코드로 PC나 모바일 장치의 위치가 변할 때마다 변경된 위치 정보를 알아낼 수 있다.
5. HTML5에서 제공하는 백그라운드 기능의 웹 워커 개념을 이해한다.
6. 자바스크립트로 웹 워커 API를 활용하여 백그라운드 작업을 만들 수 있다.

오디오/비디오 제어

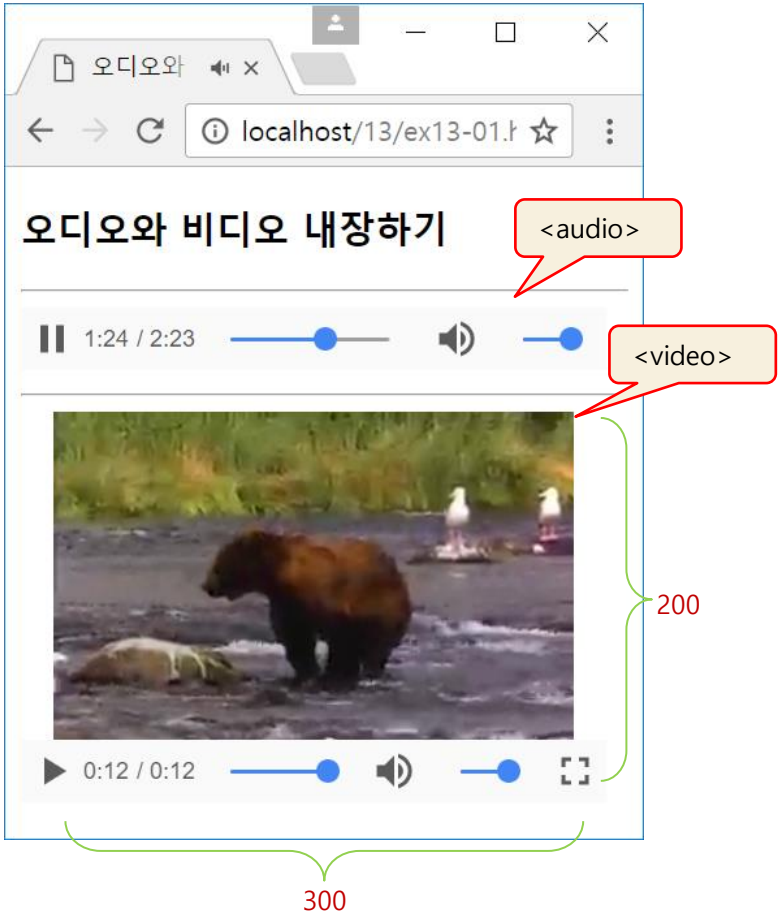
- HTML5의 오디오/비디오 제어
  - <audio>, <video> 태그
    - HTML5에서 플러그인의 도움 없이 오디오/비디오 삽입
  - 자바스크립트 코드로 미디어 재생/중단/ 등 미디오 제어
- <audio>와 <video> 태그

```
<audio id="audio" src="media/EmbraceableYou.mp3"
   autoplay loop controls>
   웹 브라우저가 audio 태그를 지원하지 않습니다.
</audio>
```


```
<video id="video" width="300" height="200"
   autoplay controls>
   <source src="media/bear.mp4" type="video/mp4">
      웹 브라우저가 video 태그를 지원하지 않습니다.
</video>
```

예제 13-1 오디오/비디오를 가진 웹 페이지


```
<!DOCTYPE html>
<html>
<head><title>오디오와 비디오 내장 페이지</title>
</head>
<body>
<h3>오디오와 비디오 내장하기</h3>
<hr>
<audio id="audio" src="media/EmbraceableYou.mp3"
        autoplay loop controls>
   웹 브라우저가 audio 태그를 지원하지 않습니다.
</audio>
<hr>
<video id="video" width="300" height="200"
        autoplay controls>
   <source src="media/bear.mp4" type="video/mp4">
   웹 브라우저가 video 태그를 지원하지 않습니다.
</video>
</body>
</html>
```



자바스크립트로 오디오 제어

- <audio> 태그에 로드된 오디오 제어

```
<audio id="audio" src="media/EmbraceableYou.mp3" autoplay loop controls>...</audio>
```

  - 오디오 DOM 객체 알아내기

```
var audio = document.getElementById("audio");
```

  - 오디오 재생 및 일시 중지

```
audio.play(); // 재생. 중지된 이후부터 재생
audio.pause(); // 일시 중지
```

  - 오디오 처음부터 재생

```
audio.load(); // src에 지정된 오디오 데이터 로드
audio.play(); // 처음부터 재생
```

  - 오디오 음량 제어와 음소거

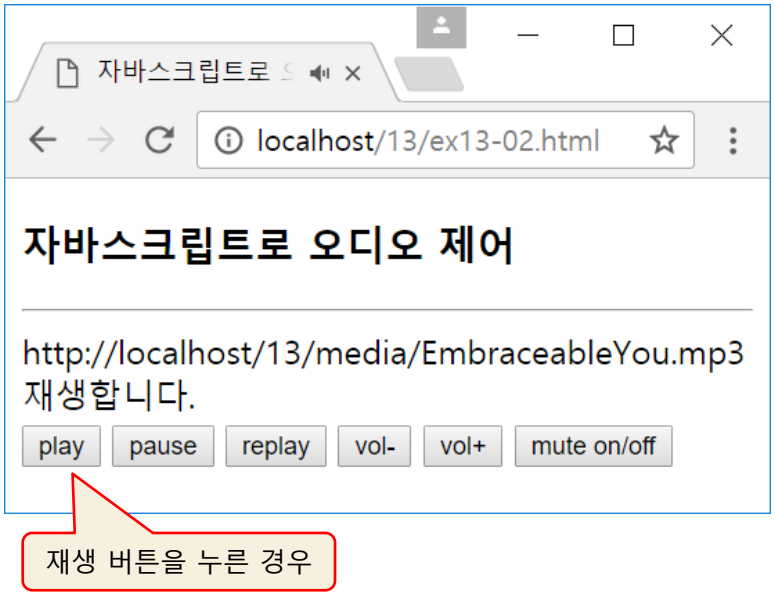
```
audio.volume += 0.1; // 0.1 만큼 음량 증가
audio.muted = true; // 음소거. 음량(volume) 변경 없음
```

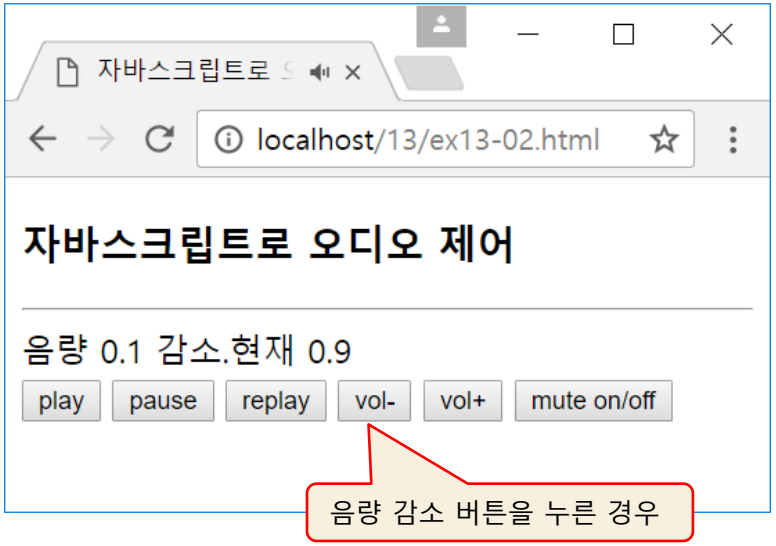
예제 13-2 자바스크립트로 오디오제어기 만들기


```
<!DOCTYPE html>
<html>
<head><title>자바스크립트로 오디오 제어</title></head>
<body>
<h3>자바스크립트로 오디오 제어</h3>
<hr>
<audio id="audio" src="media/EmbraceableYou.mp3"></audio>
<div id="msg">이곳에 오디오 제어 메시지 출력</div>
<button id="play" onclick="control(event)">play</button>
<button id="pause" onclick="control(event)">pause</button>
<button id="replay" onclick="control(event)">replay</button>
<button id="vol-" onclick="control(event)">vol-</button>
<button id="vol+" onclick="control(event)">vol+</button>
<button id="mute on/off" onclick="control(event)">mute on/off</button>
```

```
<script>
var div = document.getElementById("msg");
var audio = document.getElementById("audio");
function control(e) {
    var id = e.target.id;
    if(id == "play") { // play 버튼 클릭
        audio.play(); // 재생
        div.innerHTML = "재생중입니다.";
    }
    else if(id == "pause") { // pause 버튼 클릭
        audio.pause(); // 일시 중지
        div.innerHTML = "일시중지되었습니다.";
    }
    else if(id == "replay") { // replay 버튼 클릭
        audio.load(); // src에 지정된 미디어 다시 로딩
        audio.play(); // 처음부터 다시 재생
        div.innerHTML = audio.src + "를 처음부터 재생합니다.";
    }
    else if(id == "vol-") { // vol- 버튼 클릭
        audio.volume -= 0.1; // 음량 0.1 감소
        if(audio.volume < 0.1) audio.volume = 0;
        div.innerHTML = "음량 0.1 감소." + "현재 " +    audio.volume;
    }
    else if(id == "vol+") { // vol+ 버튼 클릭
        audio.volume += 0.1; // 음량 0.1 증가
        if(audio.volume > 0.9) audio.volume = 1.0;
        div.innerHTML = "음량 0.1 증가." + "현재 " + audio.volume;
    }
    else if(id == "mute on/off") { // mute on/off 버튼 클릭
        audio.muted = !audio.muted; // 음소거 토글
        if(audio.muted) div.innerHTML = "음소거";
        else div.innerHTML = "음소거 해제";
    }
}
</script>
</body></html>
```

예제 13-2 실행 결과


자바스크립트로
localhost/13/ex13-02.html
자바스크립트로 오디오 제어
http://localhost/13/media/EmbraceableYou.mp3 재생합니다.
play
pause
replay
vol-
vol+
mute on/off
재생 버튼을 누른 경우


자바스크립트로
localhost/13/ex13-02.html
자바스크립트로 오디오 제어
음량 0.1 감소.현재 0.9
play
pause
replay
vol-
vol+
mute on/off
음량 감소 버튼을 누른 경우

비디오 제어

- <video> 태그에 로드된 비디오 제어

```
<video id="video" width="320" height="240" autoplay controls>...</video>
```

  - 비디오 DOM 객체 알아내기

```
var video = document.getElementById("video");
```

  - width, height와 videoWidth, videoHeight 프로퍼티
    - width, height : <video> 태그의 width, height 속성 반영
    - videoWidth, videoHeight : 비디오 미디어의 화면 해상도
  - loadedmetadata 이벤트
    - 비디오 파일의 로드 완료시, video 객체에 loadedmetadata 이벤트 발생
    - 예) 비디오의 해상도 알아내기. 비디오가 로드되어야 비로소 videoWidth, videoHeight 프로퍼티가 정확한 값을 가짐

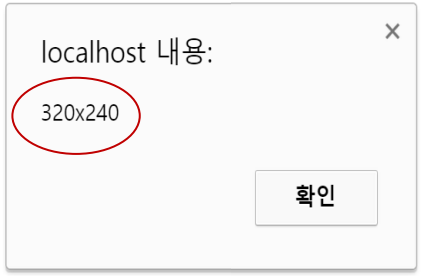
```
video.onloadedmetadata = function f(e) {
   alert(video.videoWidth + "," + video.videoHeight);
}
```

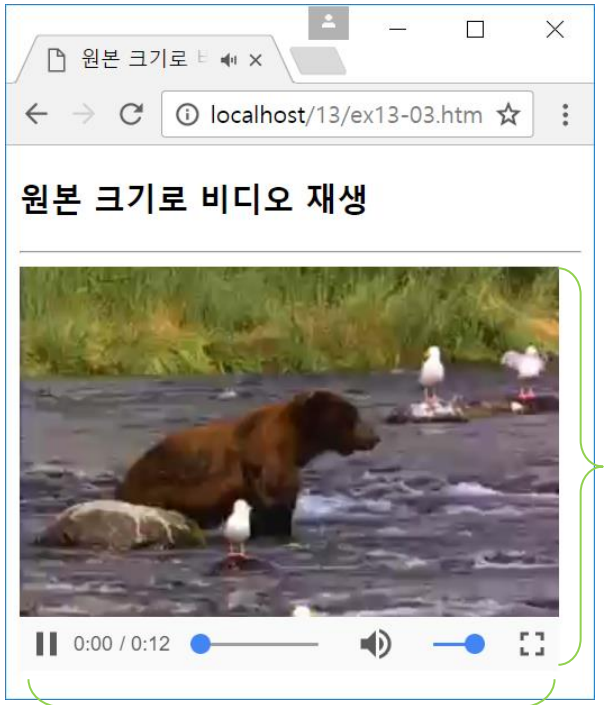
예제 13-3 비디오를 원본 크기로 재생


```
<!DOCTYPE html>
<html>
<head><title>비디오 원본 크기로 출력</title>
</head>
<body>
<h3>비디오 원본 크기로 출력</h3>
<hr>
<video id="video" width="0" height="0"
      controls autoplay>
   <source src="media/bear.mp4"  type="video/mp4">
   웹 브라우저가 video 태그를 지원하지 않습니다.
</video>
<script>
var video = document.getElementById("video");
video.onloadedmetadata = function f(e) {
   alert(video.videoWidth + "x" + video.videoHeight);
   video.width = video.videoWidth;
   video.height = video.videoHeight;
}
</script>
</body>
</html>
```

의도적인 0x0 크기

<video> 태그의 크기를 비디오의 원본 크기로 지정





오디오와 비디오의 onended 리스너

- onended 리스너
  - 오디오/비디오의 재생이 완료되었을 때 호출되는 이벤트 리스너
  - 예) 리스너 작성 사례

```
<audio id="audio" src="media/EmbraceableYou.mp3" autoplay controls></audio>

<script>
  var audio = document.getElementById("audio");
  audio.onended = function (e) {
        // ended 이벤트 처리 코드
  }
</script>
```

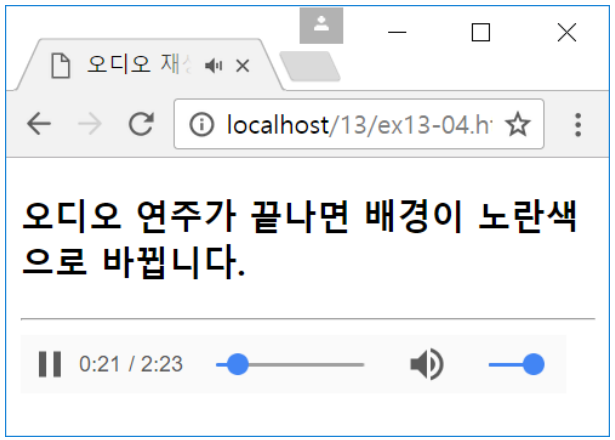
  - loop 속성이 설정되면 onended 이벤트 리스너 호출되지 않음

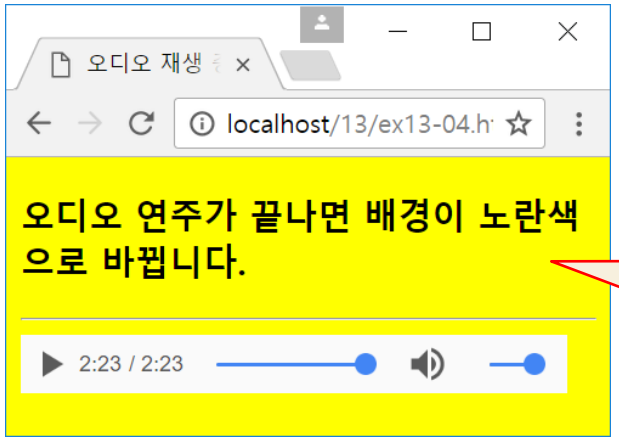
```
<audio arc="..." loop> <!-- loop 속성이 있으면 ended 이벤트 발생하지 않음 -->
```

예제 13–4 오디오 재생이 끝나면 웹 페이지를 노란색으로 변경

onended 리스너를 활용하여 오디오 재생이 끝나면 전체 배경을 노란색으로 변경하라.

```
<!DOCTYPE html>
<html>
<head><title>오디오 재생 종료 ended 이벤트 받기</title></head>
<body>
<h3>오디오 연주가 끝나면 배경이 노란색으로 바뀝니다.</h3>
<hr>
<audio id="audio" src="media/EmbraceableYou.mp3" autoplay controls></audio>
<script>
   var audio = document.getElementById("audio");
   audio.onended = function (e) { document.body.style.backgroundColor="yellow"; }
</script>
</body>
</html>
```





미디어 소스 변경/미디어 로드

- 현재 재생 중인 미디어 변경
  - 다음 3 단계 필요
    - audio.src="media/Aegukga.mp3"; // 새로운 미디어 지정
    - audio.load(); // src에 지정된 미디어 새로 로딩. 생략 가능
    - audio.play(); // 로딩된 미디어 재생

위치 정보 서비스

- HTML5의 위치 정보 서비스란?
  - 컴퓨터/모바일 장치의 위도와 경도를 자바스크립트 코드에게 공급
- geolocation 객체
  - 위치 정보 서비스를 제공하는 자바스크립트 객체

```
navigator.geolocation, window.navigator.geolocation
```

  - 위치 정보 서비스 2가지
    - 현재 위치 서비스 : 요청 시 현재 위치를 알려주는 서비스
    - 반복 위치 서비스 : 위치가 변경될 때마다 반복하여 알려주는 서비스

메소드	설명
getCurrentPosition()	현재 위치 얻기
watchPosition()	위치가 변경될 때마다 알려주는 반복 위치 서비스 시작
clearWatch()	반복 위치 서비스 중단

  - 브라우저의 위치 정보 서비스 지원 여부

```
if(navigator.geolocation) {
      // 브라우저가 위치 정보 서비스를 제공한다.
}
```

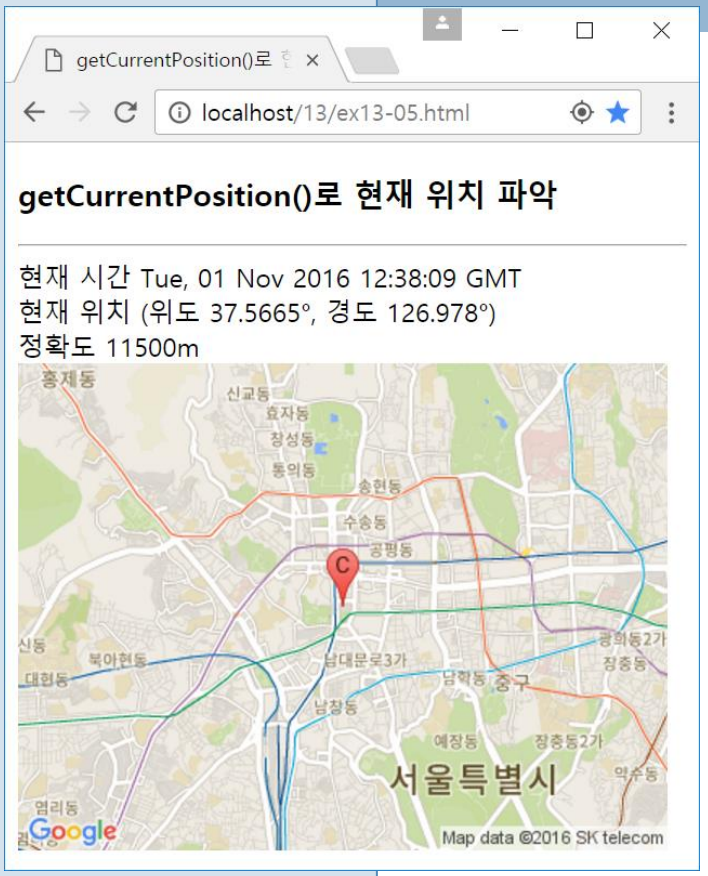
현재 위치 얻기

- 현재 위치 얻기
  - getCurrentPosition() 메소드 호출
    - getCurrentPosition()은 호출 즉시 현재 위치를 리턴하는 것이 아님
    - 위치가 파악되면 호출될 콜백 함수 positionCallback(Position) 등록

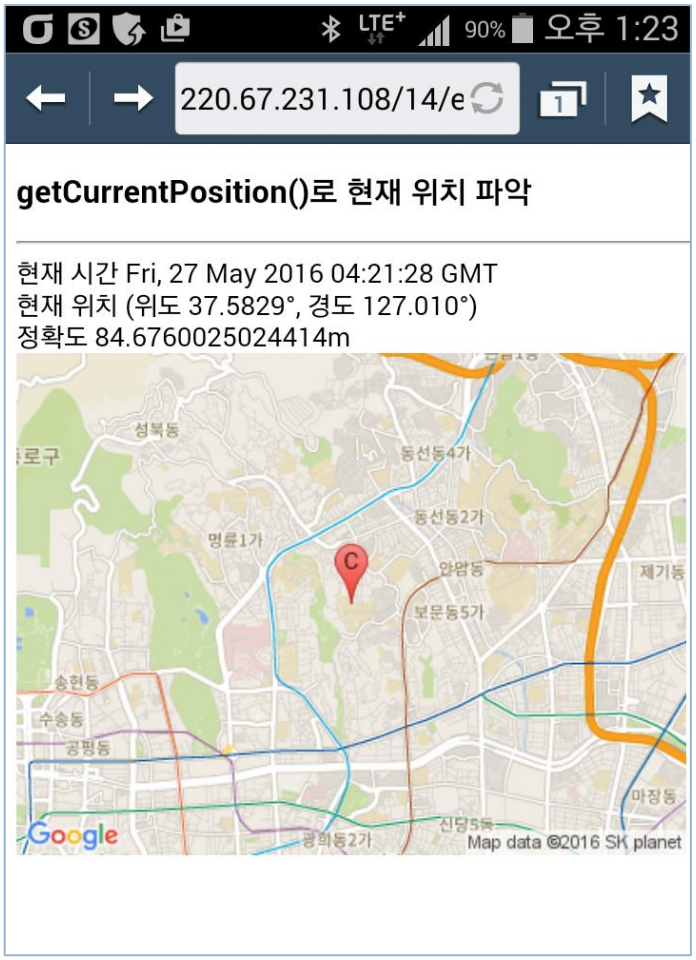
```
navigator.geolocation.getCurrentPosition(found); // found()를 콜백 함수로 등록
   ...
// 위치가 파악되면 found() 호출, 위치 정보가 있는 position 객체가 매개 변수로 전달
function found(position) {
   var lat = position.coords.latitude; // 위도
   var lon = position.coords.longitude; // 경도
   alert("현재위치(" + lat + ", " + lon + ")");
}
```

예제 13-5 getCurrentPosition()로 현재 위치파악


```
<!DOCTYPE html>
<html>
<head><title>getCurrentPosition()로 현재 위치 파악</title></head>
<body>
<h3>getCurrentPosition()로 현재 위치 파악</h3>
<hr>
<div id="msg">이곳에  위치 정보 출력</div>
<div id="map"></div>
<script>
if(!navigator.geolocation)
    alert("지원하지 않음");
else // found() 콜백 함수 등록
    navigator.geolocation.getCurrentPosition(found);

// 위치가 파악되면 found()가 호출
// 위치 정보 들어 있는 position 객가 매개 변수로 넘어온다.
function found(position) {
    var now = new Date(position.timestamp);
    var lat = position.coords.latitude; // 위도
    var lon = position.coords.longitude; // 경도
    var acc = position.coords.accuracy; // 정확도

    // 위도와 경도의 소수점 이하 자리가 너무 길어 유효 숫자 6자리로 짜름
    lat = lat.toPrecision(6); lon = lon.toPrecision(6);

    var text = "현재 시간 " + now.toUTCString() + "<br>";
    text += "현재 위치 (위도 " + lat + "°, 경도 " + lon + "°)<br>";
    text += "정확도 " + acc + "m<br>";

    document.getElementById("msg").innerHTML = text;
    var img = new Image();
    img.src = "https://maps.googleapis.com/maps/api/staticmap?center=" + lat
    + "," + lon + "&zoom=13&size=400x300&sensor=false&markers=color:red%7Clabel:C%7C"+lat +"," +lon;

    document.getElementById("map").appendChild(img); // 구글 지도 이미지를 div의 자식으로 붙임
}
</script></body></html>
```



삼성 갤럭시 탭에서도 잘 동작



- Google Geolocation 서비스를 받기 위해서는 API_KEY를 Google Developers Console에서 발급받아 사용해야 함.
- 현재 대부분 모바일 장치에서는 API_KEY가 받아져 있기 때문에 이 예제는 모바일 단말기에서는 잘 작동함

반복 위치 서비스

- 위치 변경 시마다 현재 위치 얻기
  - watchPosition() 호출
    - 위치가 변경될 때마다 호출되는 콜백 함수 등록
    - watchPosition()의 리턴 값 : 반복 위치 서비스 id

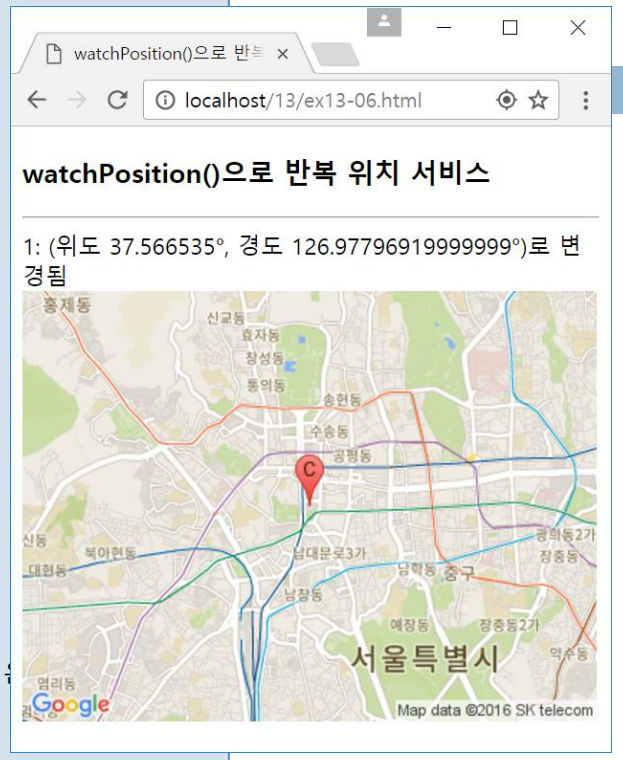
```
var watchID = navigator.geolocation.watchPosition(changed); // changed()를
                                                    //  콜백 함수로 등록하고, 반복 위치 서비스 시작
...
// 위치가 바뀌면 changed() 호출, 위치 정보가 있는 position 객체가 매개 변수로 전달
function changed(position) {
   var lat = position.coords.latitude; // 변경된 위도
   var lon = position.coords.longitude; // 변경된 경도
   alert("(" + lat + ", " + lon + ") 위치로 변경됨");
}
```

- 반복 위치 서비스 중단
  - clearWatch() 호출
    - watchPosition()가 리턴한 id로 반복 위치 서비스 중단

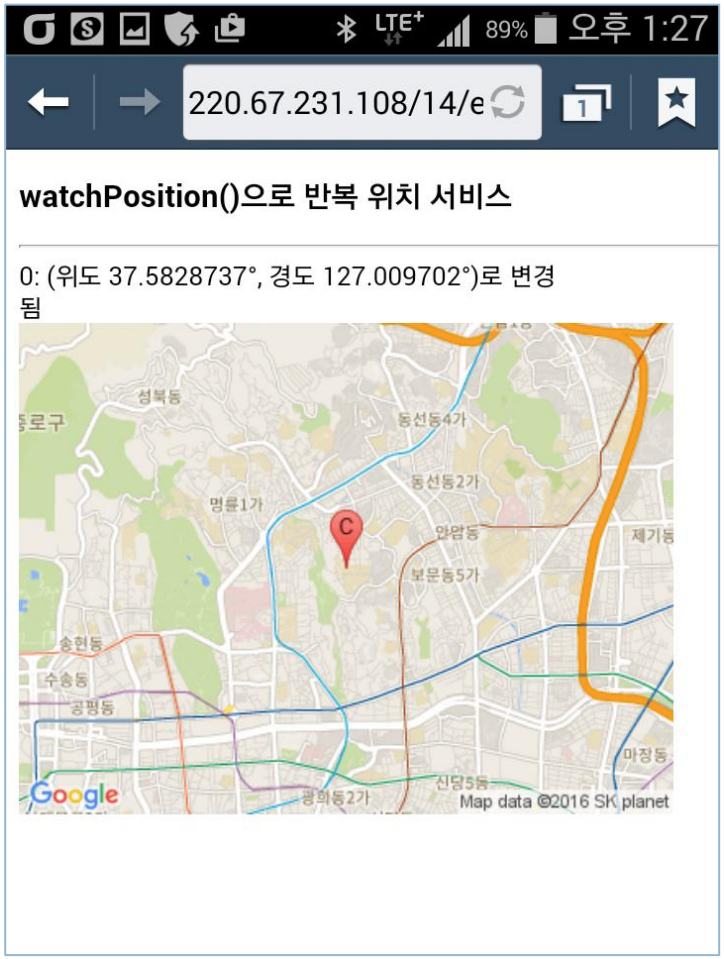
```
navigator.geolocation.clearWatch(watchID); // watchID의 반복 위치 서비스 중단
```

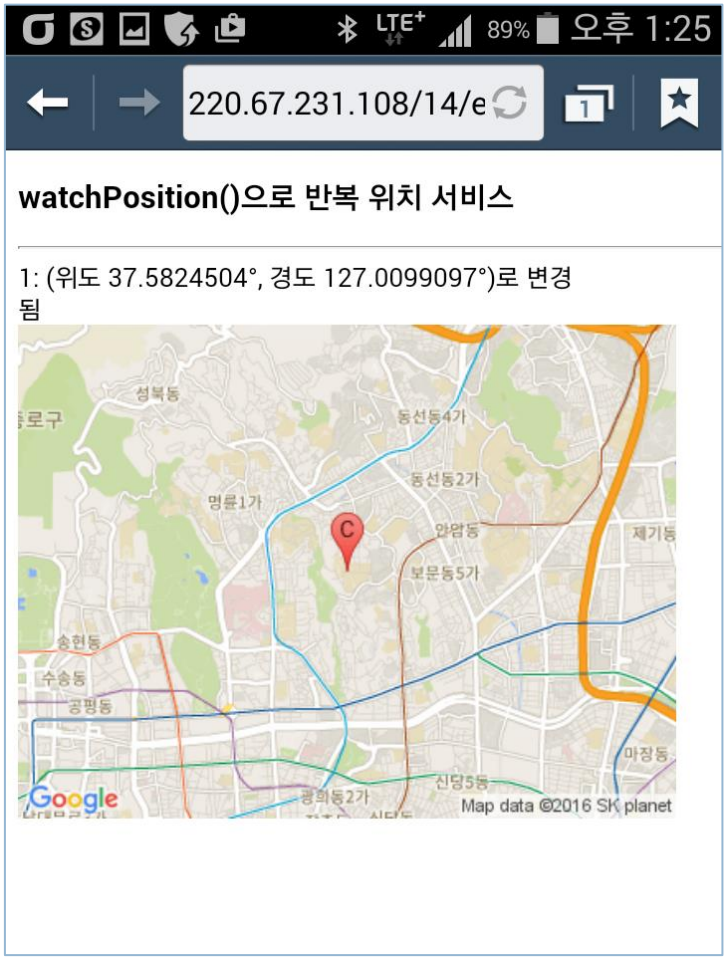
예제 13-6 watchPosition()으로 반복 위치서비스


```
<!DOCTYPE html>
<html>
<head><title>watchPosition()으로 반복 위치 서비스</title></head>
<body>
<h3>watchPosition()으로 반복 위치 서비스</h3>
<hr>
<div id="msg">이곳에  위치 정보 출력</div>
<div id="map"></div>
<script>
if(!navigator.geolocation)
    alert("지원하지 않음");
else {
    var options = { // 3 개의 값을 가진 전역 객체. watchPosition()의 마지막 매개 변수로 전달
        enableHighAccuracy: false,
        timeout: 5000,
        maximumAge: 0 };

    var img = new Image();
    var count=0;
    var watchID;
    // changed() 콜백 함수 등록하고, 반복된 위치 서비스를 시작시킨다.
    watchID = navigator.geolocation.watchPosition(changed, null, options);
}
//위치가 바뀌면 changed()가 호출되고, 위치 정보가 들어 있는 position 객체가 매개 변수로 넘어
function changed(position) {
    if(count == 5) { // clearWatch() 테스트를 위해 5번만 서비스
        navigator.geolocation.clearWatch(watchID); // 반복 서비스 종료
        document.getElementById("msg").innerHTML = "위치 서비스 종료";
        return;
    }
    var lat = position.coords.latitude; // 변경된 위도
    var lon = position.coords.longitude // 변경된 경도
    var text = count + ": (위도 " + lat + "°, 경도 " + lon + "°)로 변경됨<br>";
    document.getElementById("msg").innerHTML = text; // 위치 정보 출력

    // 지도 이미지 갱신
    img.src = "https://maps.googleapis.com/maps/api/staticmap?center=" + lat
        + "," + lon + "&zoom=13&size=400x300&sensor=false&markers=color:red%7Clabel:C%7C"+lat      +","
+lon;
    if(count == 0) // 처음이면 구글 지도 이미지 부착
        document.getElementById("map").appendChild(img); // 지도 이미지 부착
    count++; // 갱신 회수 증가
}
</script></body></html>
```



삼성 갤럭시 탭에서도 반복 위치 서비스 작동





웹 워커

- 웹 워커(Web Workers)란?
  - 백그라운드 태스크를 만드는 기능
    - 자바스크립트 코드를 백그라운드에서 별도로 실행시킬 수 있는 HTML5 표준 기능
    - 백그라운드 태스크를 워커 태스크라고 부름
  - 실행 시간이 긴 계산 작업에 적합
    - 워커 태스크는 윈도우와 사용자 인터페이스 사용 불가능
- 웹 워커의 특징
  - 백그라운드 태스크로 실행할 자바스크립트 코드는 파일 형태로 저장
  - 동일 도메인(same origin) 원칙 준수
    - 자바스크립트 파일은 웹 페이지와 동일한 웹 사이트에 저장
  - 로컬 컴퓨터의 웹 페이지에서는 작동하지 않음
    - 웹 서버 설치 및 작동 필요(부록 참고)

워커 객체와 워커 태스크

- 워커 태스크(Worker Task)
  - 웹 워커 기능을 이용하여 만든 백그라운드 태스크
- 워커 태스크 만들기

1. 워커 태스크를 만들 자바스크립트 코드(add1to10.js) 준비

```
// 1에서 9까지 더하고 결과 전송
var sum = 0;
for(var i=0; i<10; i++) {
   sum += i;
}
postMessage(sum); // sum을 메인 태스크에 전송
```

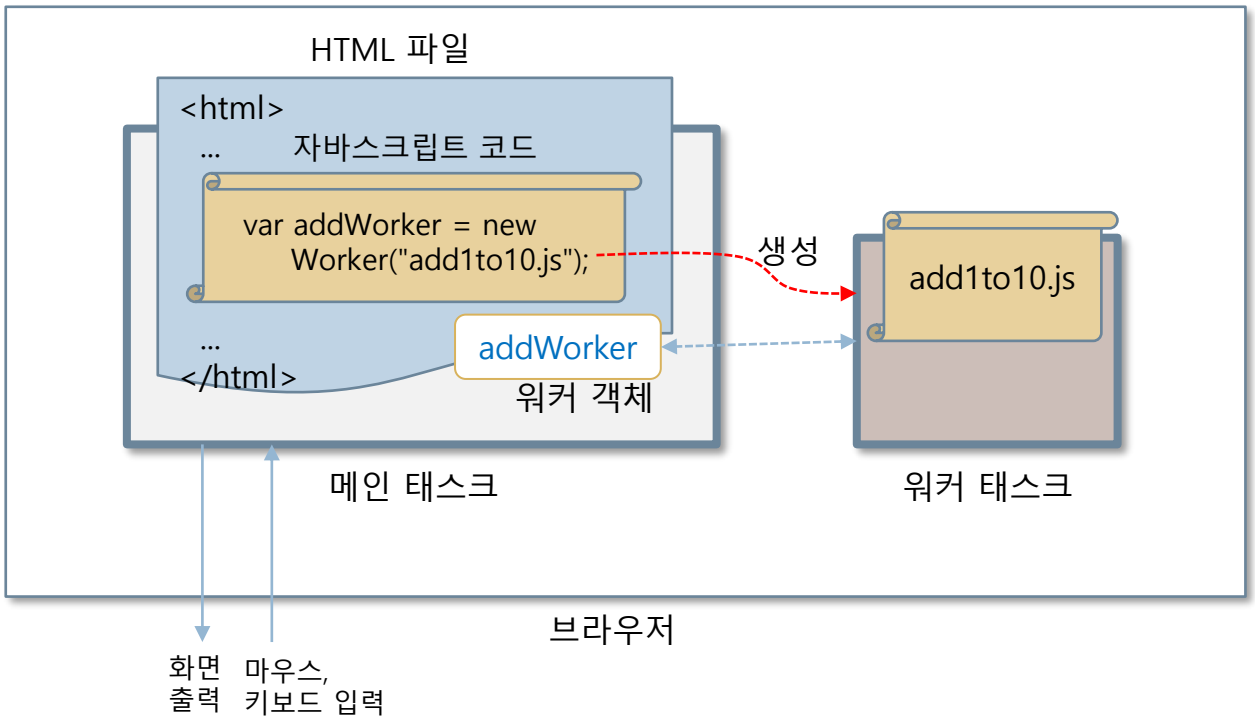
2. 워커 태스크와 워커 객체 생성

```
var addWorker = new Worker("add1to10.js");
```

   - addWorker -워커 객체
   - new Worker("add1to10.js") - 워커 태스크 생성

var addWorker = new Worker("add1to10.js");

- 워커 태스크
  - 독립적으로 실행되는 작업 단위
- 워커 객체
  - 워커 태스크로부터 이벤트와 데이터 주고 받기
  - 워크 태스크 중단 등, 워커 태스크를 제어하는 객체



브라우저의
메인 태스크와
워크 태스크로 구성된
멀티 태스킹

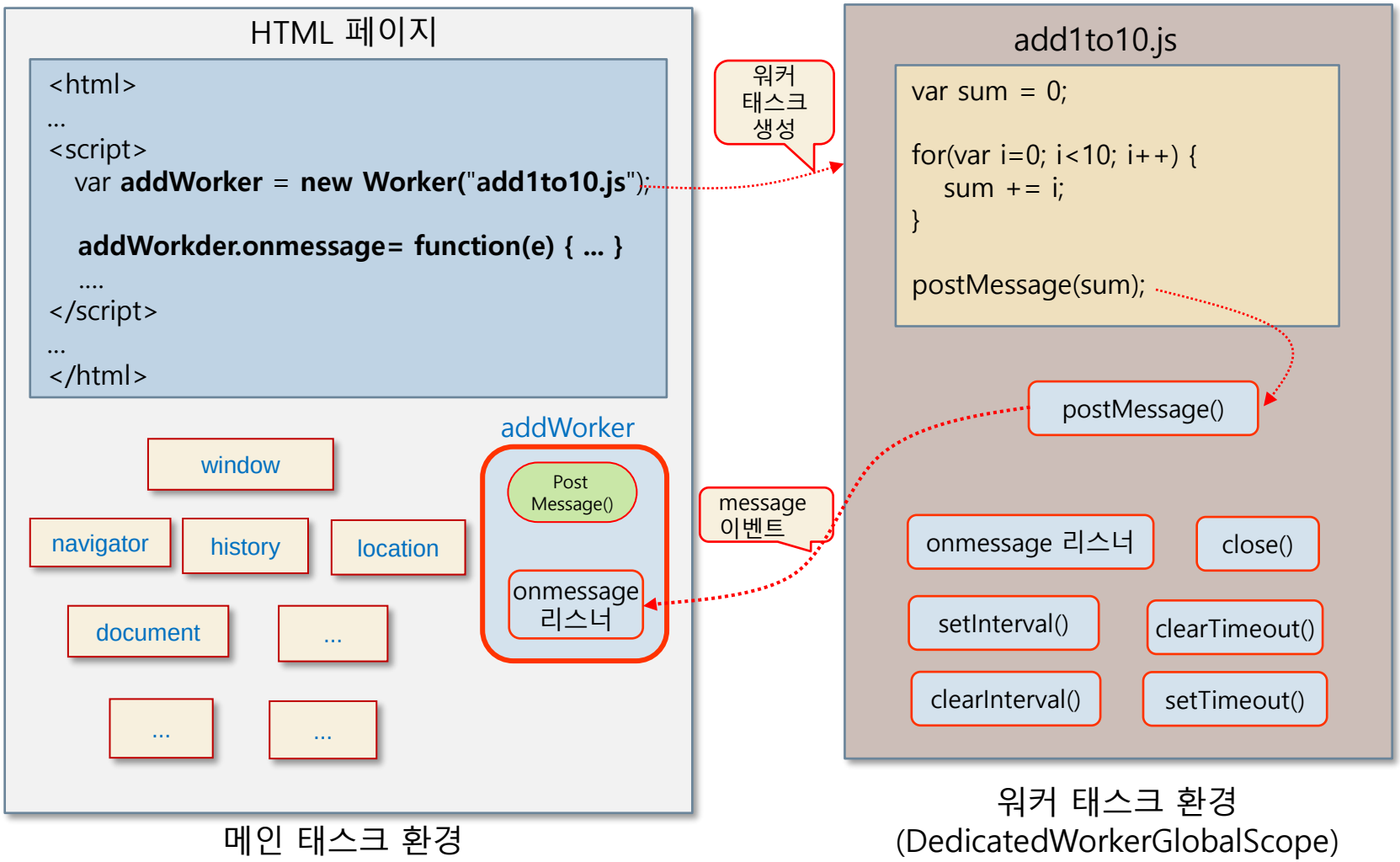
워커(Worker) 객체의 메소드와 이벤트 리스너

메소드	설명
Worker()	워커 태스크 생성
postMessage()	워커 태스크에 메시지 전송. 워커 태스크에는 message 이벤트 발생
terminate()	즉각 워커 태스크의 실행을 종료시킴

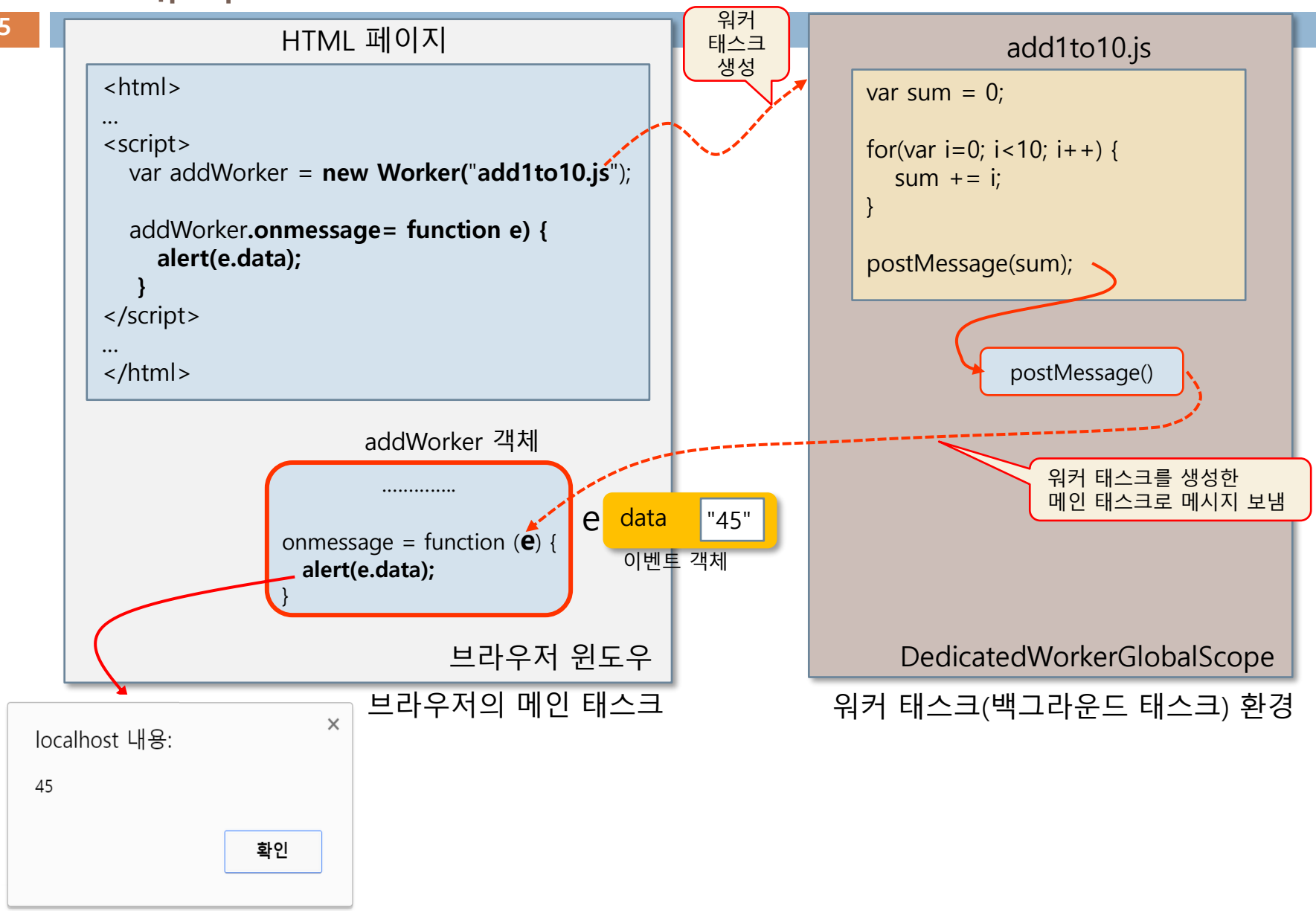
이벤트 리스너	설명
onmessage	워커 태스크로부터 발생한 message 이벤트를 받는 리스너
onerror	오류가 발생할 때 받는 이벤트 리스너

워커 태스크의 실행 환경

- 워커 태스크는 UI를 사용할 수 없는 별도의 실행 환경



워커 태스크에서 워커 객체로 message 이벤트 보내기



message 이벤트 보내는 과정

- 워커 태스크에서 postMessage() 호출
  - 매개 변수에 보내고자 하는 데이터 전달

```
postMessage(sum);        // sum = 45
```

  - 데이터를 MessageEvent 객체로 만들어 전달
    - 워커 객체에서 MessageEvent 객체를 통해 데이터 수신
    - MessageEvent 객체의 프로퍼티

프로퍼티	설명	r/w
data	전달되는 값으로서 문자열이거나 객체	r

- 워커 객체의 onmessage 리스너
  - onmessage 리스너
    - 워커 태스크가 보내는 message 이벤트를 받는 코드

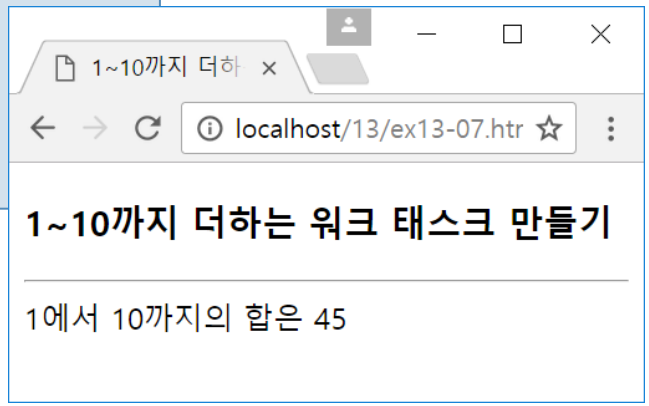
```
addWorker.onmessage = function (e) { // e에 MessageEvent 객체가 전달
   alert(e.data); // e.data는 "45"
}
```

예제 13–7 1~10까지 더하는 워커태스크 만들기


```
<!DOCTYPE html>
<html>
<head><title>1~10까지 더하는 워크 태스크 만들기</title></head>
<body>
<h3>1~10까지 더하는  워크 태스크 만들기</h3>
<hr>
<div>1에서 10까지의 합은 <span id="sum"></span></div>
<script>
  // addWorker 워커 객체 생성 및 워커 태스크 시작
  var addWorker = new Worker("add1to10.js");

  // 워크 태스크로부터 message 이벤트 수신
  addWorker.onmessage = function (e) { // e는 MessageEvent 객체
      // 이벤트 객체의 data(합) 출력
    document.getElementById("sum").innerHTML = e.data;
  }
</script>
</body>
</html>
```

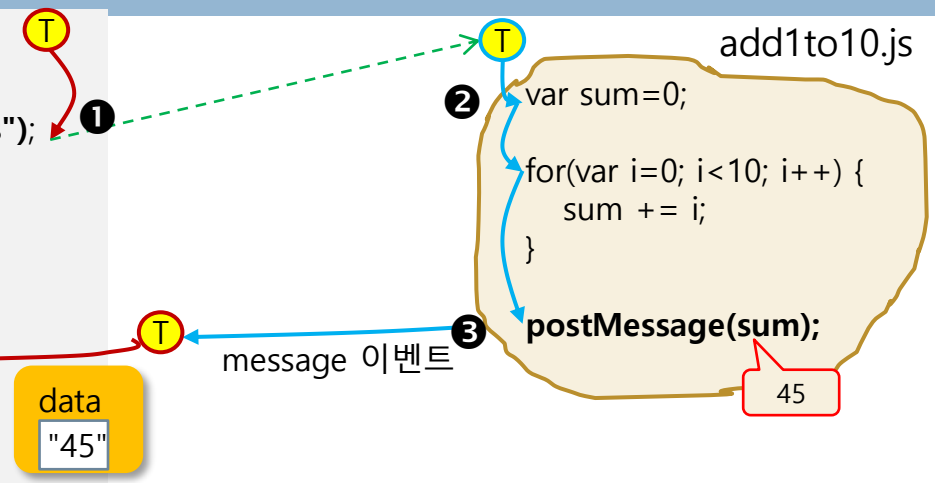
add1to10.js

```
// 1~10까지 합 계산
var sum=0;
for(var i=0; i<10; i++) {
   sum += i;
}

// 합을 메시지로 전송
postMessage(sum);
```

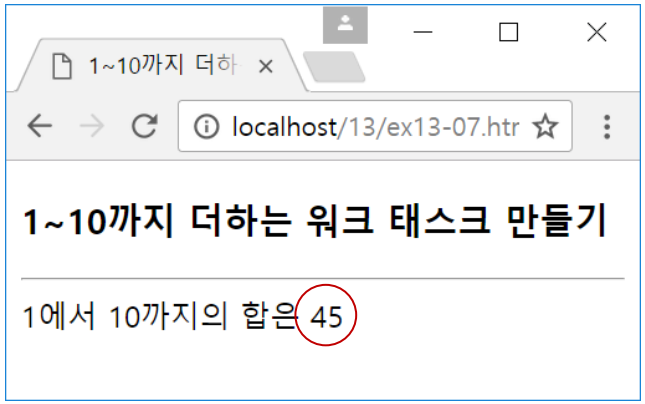


HTML 페이지

```
<script>
var addWorker = new Worker("add1to10.js");

        ... 다른 작업 진행 ...


addWorker.onmessage = function (e) {
        document.getElementById("sum")
                .innerHTML = e.data;
}
</script>
```



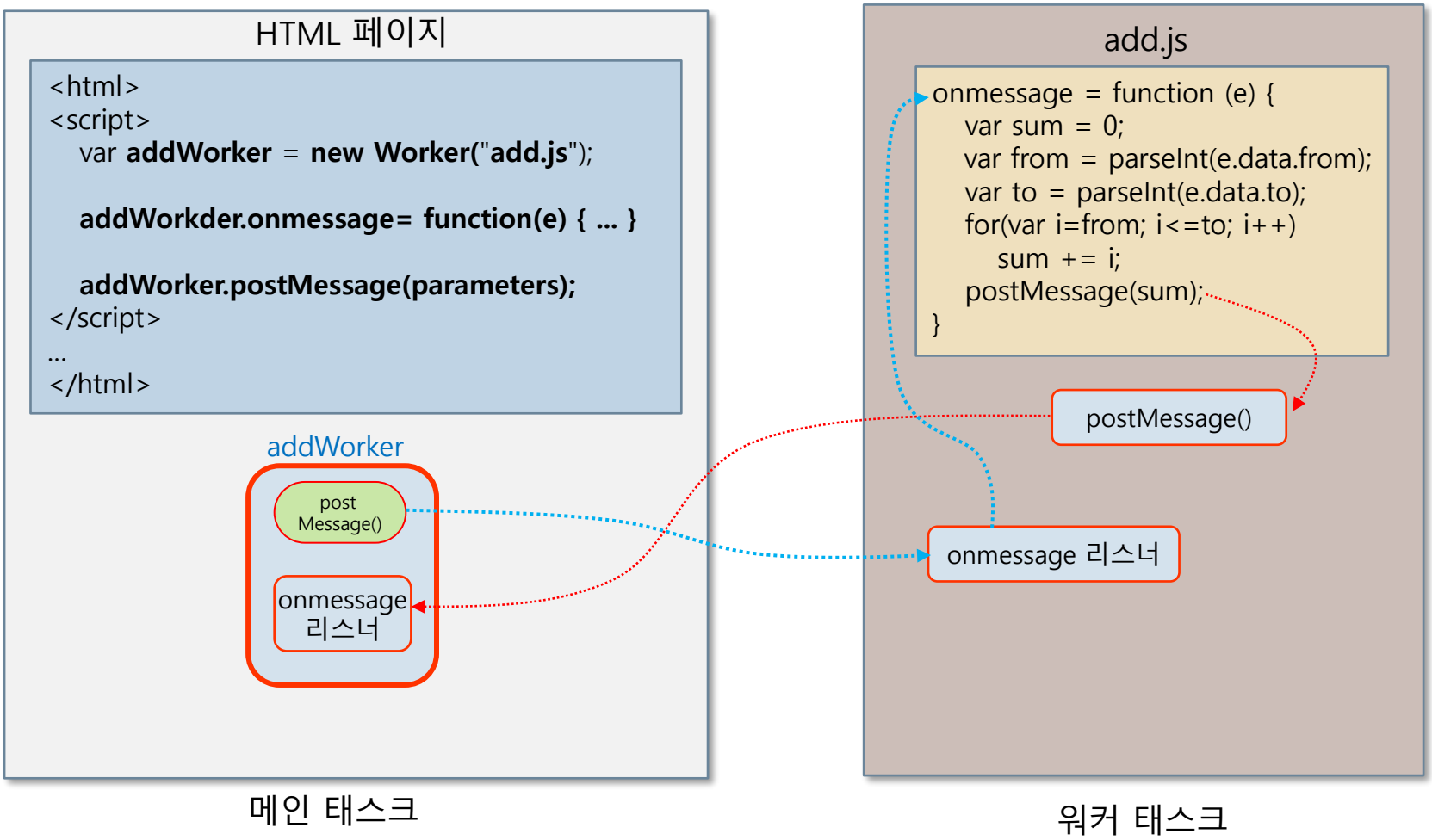
실행 결과



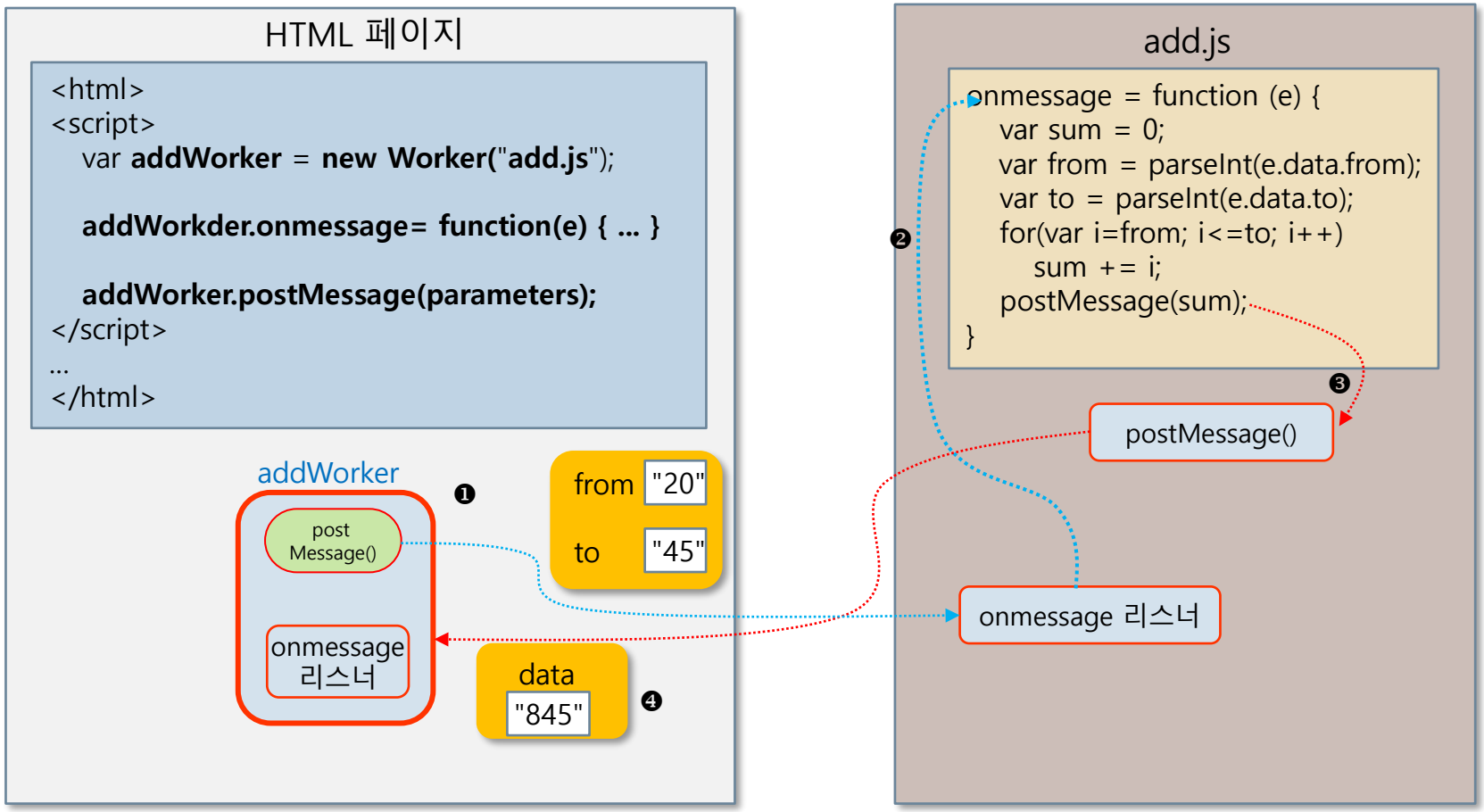
T : 브라우저 메인 태스크
T : 워커 태스크

메인 태스크에서 워커 태스크로 message 이벤트 보내기

var addWorker = new Worker("add.js");의 실행으로형성된 메인 태스크와 워커 태스크



메인 태스크와 워커 태스크 사이의 데이터 전송

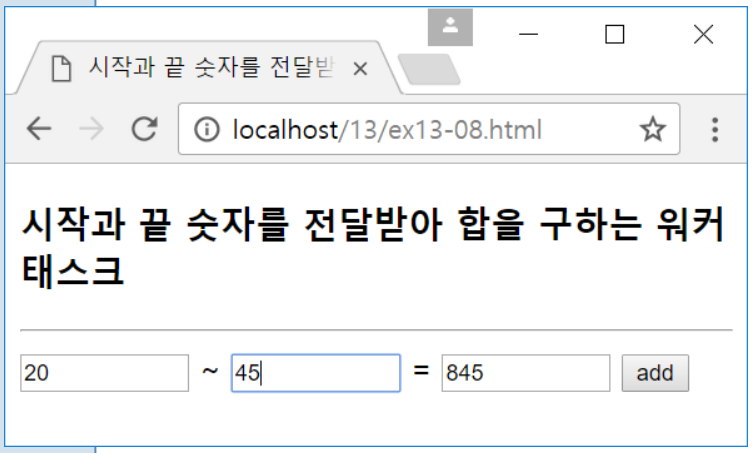


예제 13-8 워커 태스크에 시작 숫자과 끝 숫자를 보내고 합을 전달받는 코드


```
<!DOCTYPE html>
<html>
<head><title>시작과 끝 숫자를 전달받아 합을 구하는 워커 태스크</title></head>
<body>
<h3>시작과 끝 숫자를 전달받아 합을 구하는 워커 태스크</h3>
<hr>
<input id="from" type="text" size="10"> ~
<input id="to" type="text" size="10"> =
<input id="sum" type="text" size="10">
<button id="add" onclick="send()">add</button>
<script>
    var addWorker = new Worker("add.js"); // 워커 태스크 생성

    function send() { // 워크 태스크에 시작 숫자와 끝 숫자 전송
        var parameters = { // 시작 숫자와 끝 숫자로 구성된 객체
            from: document.getElementById("from").value,
            to: document.getElementById("to").value
        };
         // 시작 숫자와 끝 숫자를 담은 객체를 워커 태스크로 전송
        addWorker.postMessage(parameters);
    }

    // 워커 태스크로부터 결과를 기다리는 리스너 등록
    addWorker.onmessage = function (e) {
        // 워커 태스크로부터 전달받은 합 출력
        document.getElementById("sum").value = e.data;
    }
</script>
</body></html>
```

add.js

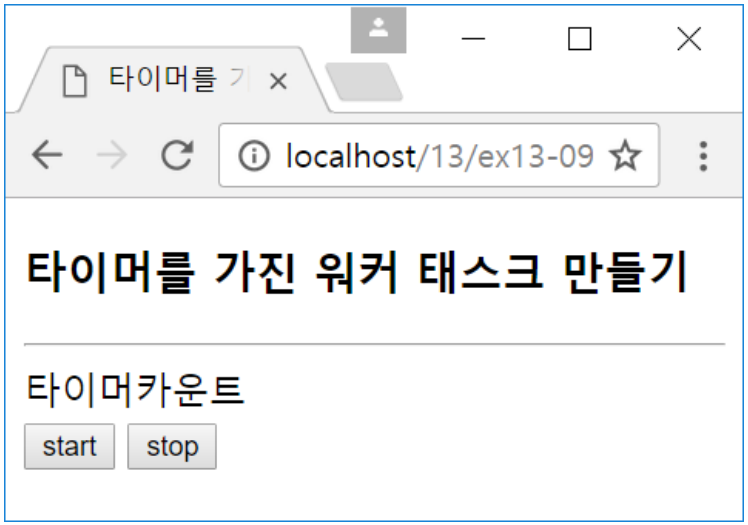
```
onmessage = function (e) {
    var sum = 0;
    var from = parseInt(e.data.from);
    var to = parseInt(e.data.to);
    for(var i=from; i<=to; i++)
        sum += i;
    postMessage(sum);
}
```



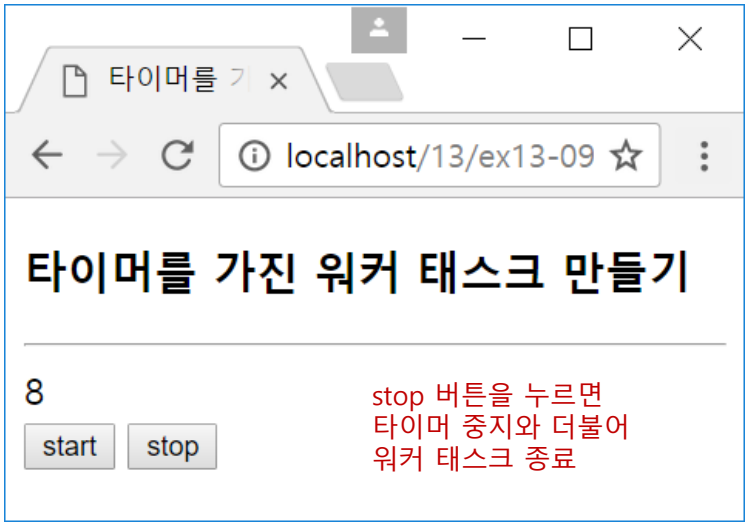
워커 태스크 종료

- 워크 태스크 스스로 종료
  - close()
- 워커 객체가 워커 태스크를 강제 종료
  - terminate() 메소드
  - 예) addWorker.terminate();
- 워커 태스크가 종료하면 워커 객체는 더 이상 워커 태스크와 message 이벤트를 주고받을 수 없다.

예제 13-9 1초 단위로 메시지를 보내는 워커 태스크 만들기



초기 화면



start 버튼을 누르면 타이머 작동

예제 13-9의 코드


```
<!DOCTYPE html>
<html>
<head><title>타이머를 가진 웹 워커 만들기</title></head>
<body>
<h3>타이머를 가진 웹 워커 만들기</h3>
<hr>
<div><span id="timer">타이머카운트</span></div>
<button id="start" onclick="start()">start</button>
<button id="stop" onclick="stop()">stop</button>
<script>
var addWorker = new Worker("timer.js"); // 워커 태스크 생성

addWorker.onmessage = function (e) {
    document.getElementById("timer").innerHTML = e.data;
}

function start() {
    addWorker.postMessage("start");
}

function stop() {
    addWorker.postMessage("stop");
}
</script>
</body>
</html>
```

timer.js

```
var count=1;
var intervalID=null; // 타이머 ID

onmessage = function (e) { // 브라우저로부터 메시지 수신
    if(e.data == "start") {
        if(intervalID != null) return; // 타이머 작동중이면 리턴
        intervalID = setInterval(myCallback, 1000); // 1초 간격 myCallback() 호출
    }
    else if(e.data == "stop") {
        if(intervalID == null) return; // 타이머 작동하지 않으면 리턴
        clearInterval(intervalID);
        close(); // 워커 태스크 종료. 더 이상 메시지 받지 않음
    }
}

function myCallback() { // 1초 간격으로 호출
    postMessage(count); // 카운트 값을 브라우저로 전송
    count++; // 카운트 값 증가
}
```