15강. HTML API [2] : 웹 워커, 웹 소켓, 위치 정보

1. 웹 워커

1.1 멀티 스레드와 웹 워커

- 하나의 응용 프로그램이 스레드라 불리는 처리 단위를 복수 개 생성하여 동시에 여러 기능을 수행하는 것을 멀티 스레드(multi-thread) 라고 한다.
- 웹 브라우저는 싱글 스레드를 지원한다. 오랜 시간 걸리는 작업이 있다면 작업이 종료될 때까지 기다려야 하고, 복잡한 처리 과정이 진행된다면, 웹 브라우저는 아마도 응답 없음 상태가 되어 버릴 것이다. 이럴 경우, 보통 웹 브라우저를 강제 종료 한 후 다시 시켜야 하는 문제가 발생할 수 있다.
- 멀티 스레드와 유사한 개념으로 HTML5에서는 웹 워커 기능이 추가되었다.
- 페이지 성능에 영향을 주지 않고, 메인이 아닌 백그라운드에로 여러 가지 복잡한 작업도 처리가 가능하게 되었다.
- 웹 워커에서 사용할 수 있는 기능과 접근할 수 없는 기능

사용 가능한 기능	접근할 수 없는 기능
navigator객체	
location 객체 (읽기 전용)	
XMLHttpRequest	DOM (스레드 비보호)
setTimeout() / clearTimeout()	window 객체
setInterval() / clearInterval()	document 객체
애플리케이션 캐시	parent 객체
ImportScripts()로외부 스크립트 가져오기	
다른 웹 워커의 생성	

1.2 전용 워커

- 메인에서 동작하는 UI 스레드와는 별개로 백그라운드에서 여러 개의 워커들이 각각의 기능을 하며 처리 동작을 하고 있는 형태로 구성한다.
- 메인 UI 문서에서 워커 역할을 하는 자바스크립트 파일을 인수로 지정하여 워커 생성자를 호출하여 워커를 생성한다.

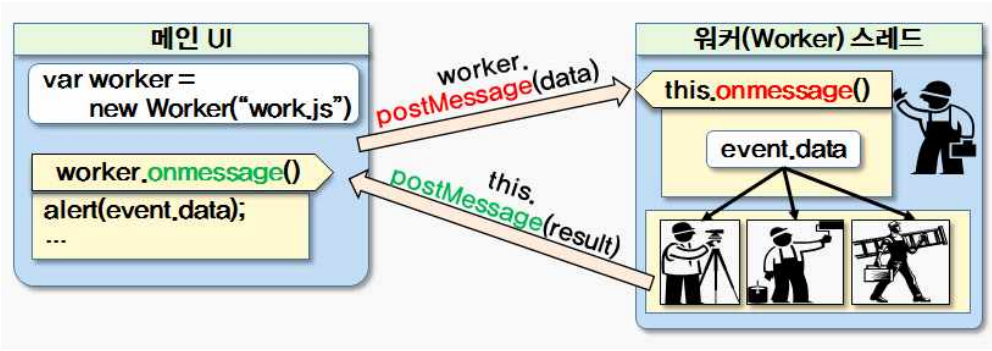
```
var 변수 = new Worker("자바스크립트파일.js");
```

```
if ( window.Worker ) {
    var worker = new Worker("calc.js");
}
```

```
if ( typeof(Worker)!=="undefined" ) {
    var worker = new Worker("calc.js");
}
else {
    // 웹 워커 미지원...
}
```

1.2.1 메시지 전달

- 워커에서 생성한 함수와 변수는 외부에서 호출 불가하므로 데이터를 보내기 위해서는 postMessage() 메서드를 사용하고, 메서드에 전달되는 데이터를 받기 위해서는 onmessage 이벤트 핸들러를 통해서 받아야 한다.



1.2.2 워커의 생성과 메시지 전달

```
if (window.Worker) {                          메인 UI 스레드
  //워커 생성
  var worker = new Worker("work.js");
  //워커로 데이터 전달
  worker.postMessage("task=job1");
  //워커에서 전달 받은 데이터 처리
  worker.onmessage = function(event) {
    alert(event.data);
  };
  //워커 종료
  worker.terminate();
}
worker.addEventListener('message', function(event) {
    alert(event.data);
}, false);
```

```
워커("work.js")
this.onmessage = function(event) {
  //메시지 처리
  var job = event.data;
  if ( job == "task=job1" ) {  job1();  }
  else { job2(); }
  //self.close()  : 워커 종료
};
function job1() {
  //결과 데이터 전송
  this.postMessage("작업 완료");
}
```

1.2.3 워커의 오류 처리

- 오류가 발생하면 Worker 객체에서 error 이벤트가 발생한다.

```
worker.onerror = function(event) {
var err_msg = event.message 오류 내용을 텍스트로 반환 + " ₩n" +
event.filename 오류가 발생한 파일의 URL을 완전한 URL(http://로 시작)로 반환 +"의"+
event.lineno 오류가 발생한 줄의 번호를 반환 + " 번째 줄에서 오류 발생"
  alert(err_msg);
}
```

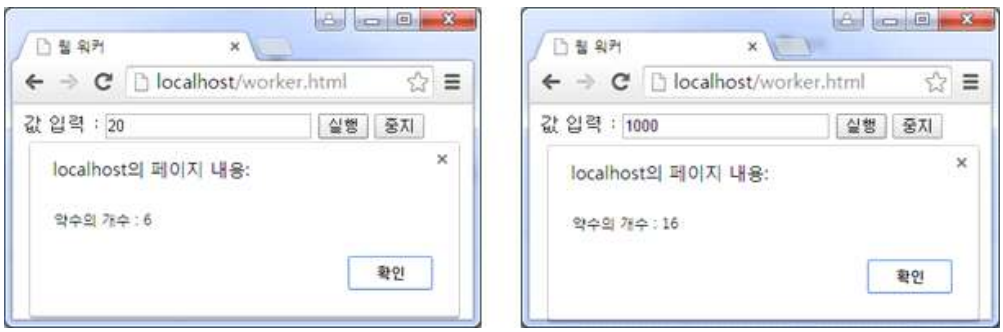
예제 웹 워커

메인 UI 스레드

```
<!DOCTYPE html><html><head>
  <script>
  var worker;
  function calc()  { //실행 부분
    var num = document.getElementById("num").value;
    worker = new Worker("calc.js");
    worker.onmessage = function(event) { //work에서 받은 메시지 처리
   alert("약수의 개수 : " + event.data); };
    worker.onerror = function(event) {alert("에러 : " + event.message);     };
    worker.postMessage(num); //메시지 전달
  }
  function stop()  {     worker.terminate();  } //중지 부분
  </script>
</head>
<body>
  값 입력 : <input type="text" id="num">
  <input type="button" onclick="calc()" value="실행">
  <input type="button" onclick="stop()" value="중지">
</body>
</html>
```

워커("calc.js")

```
onmessage = function(event) {     var num = event.data;     var total = 0;
    var i
    for ( i=num i>0; i-- ) {
        if ( num % i == 0 )
        total++;
    }
    postMessage(total); //메시지 전달
};
```



웹 서버에서 동작화면

1.3 공유 워커

- 여러 개의 워커 객체가 하나의 백그라운드 프로세스를 공유해서 사용할 수 있도록 하고 있다. 하나의 워커 객체가 백그라운드 프로세스와 일대일 대응한다.
- 공유 워커 객체를 생성하기 위해서는 SharedWorker() 생성자를 호출해야 한다. 이렇게 생성된 공유 워커는 백그라운드 프로세스를 공유하게 된다.

```
var 변수 = new SharedWorker(자바스크립트 파일명, 워커 이름);
var worker1 = new SharedWorker("calc.js", "share");
var worker2 = new SharedWorker("calc.js", "share");
```

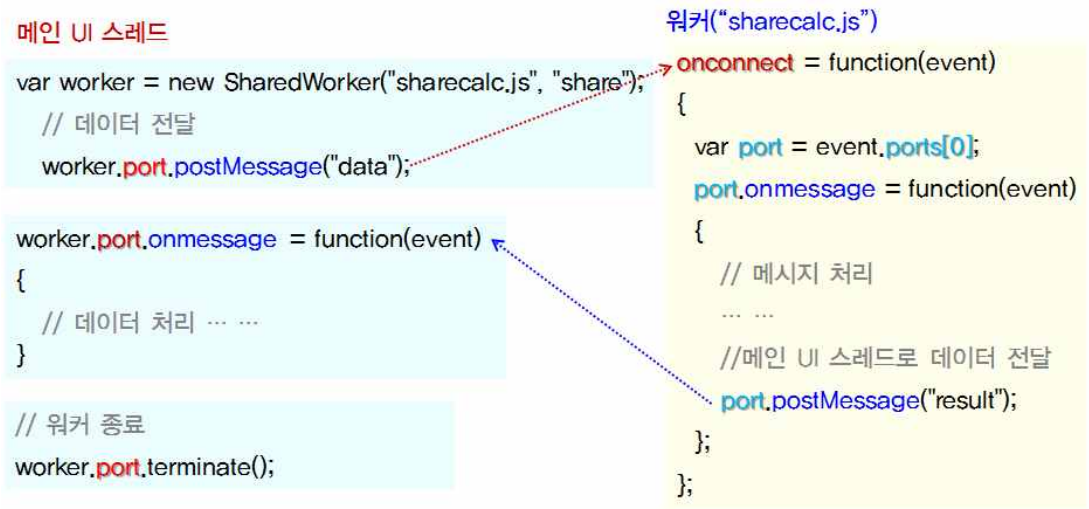
1.3.1 공유 워커의 메시지 전달

- port 속성은 메인 UI 문서와 공유 워커와의 통신을 위한 채널이다. 생성된 워커 객체의 port 속성을 사용해서 postMessage() 메서드를 호출해야 한다.
- 공유 워커에서 데이터를 전달받기 위해 사용 onconnect 이벤트를 통해서 전달받아야 한다.

```
onconnect = function(event) {
  // 메인 UI 스레드와 연결되는 포트 정보 확인
  // postMessage()로 전달된 데이터 수신
};
```

- 메인 UI 스레드와 연결되는 포트 정보를 확인할 수 있다. ports 속성은 배열 형태로 되어 있기 때문에 ports 배열의 첫 번째 값을 사용하여 포트를 확인 할 수 있다.
- postMessage()로 전달된 데이터를 공유 워커에서 수신한다.

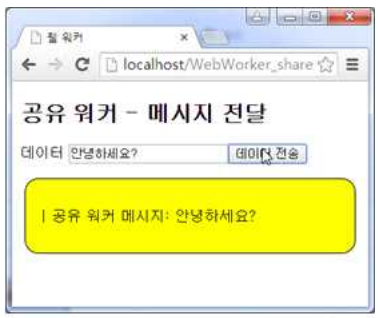
```
port.onmessage = function(event) {
  // 메시지 처리
    … …
  // 메인 UI 스레드로 데이터 전달
  port.postMessage("result");
};
```

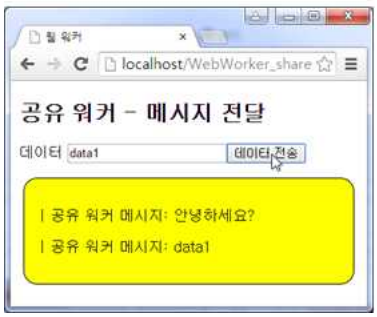


예제 공유 워커

메인 UI 스레드

```
<!DOCTYPE html><html><head>
<style>
div { background: yellow; padding: 15px; margin: 5px;
        border: 1px solid black; border-radius: 15px;    }
</style>
<script>
var worker;
window.addEventListener("load", function() {
   worker = new SharedWorker("shared_work.js"); //공유 워커 생성
   worker.port.onmessage = function(event) {  print(event.data); }//메시지를 수신
}, false);
function send() { //공유 워커로 메시지 전송
   worker.port.postMessage(document.getElementById("data").value);
}
function print(msg) {  //메시지를 출력한다.
   var p = document.createElement("p");
   p.textContent = msg;
   document.getElementById("result").appendChild(p);
}
</script>
</head>
<body>
<p>데이터 <input id="data" type="text" />
<button onclick="send()">데이터 전송</button></p>
<div><output id="result"></output></div>
</body>
</html>
```





웹에서 결과 화면

워커("calc.js")

```
self.onconnect = function(event) {
    var port = event.ports[0];
    port.onmessage = function(ev) {
        port.postMessage("| 공유 워커 메시지: " + ev.data );
    };
};
```

1.4 워커에서 사용 가능한 API

- 자바 스크립트 파일을 로드하는 방법

```
self.importScripts("lib1.js");
self.importScripts("lib2.js");
self.importScripts("lib1.js", "lib2.js");
```

- 브라우저 정보를 얻기

```
self.navigator.appName : 브라우저의 이름 반환
self.navigator.appVersion : 브라우저의 버전 반환
self.navigator.platform : 운영체제의 이름 반환
self.navigator.userAgent : 웹 서버에 보낼 User-Agent 헤더의 값을 반환
self.navigator.onLine : 네트워크에 접속할 수 없으면 false 반환
```

- 워커 자바스크립트 파일의 URL 정보
  ▸다음의 URL이 있다고 가정한다면
  http://dev.w3.org/html5/workers/work.js?arg1=a&arg2=b#apis-available-to-workers

속성	분해 내용	의미
self.location.protocol	http:	프로토콜
self.location.host	dev.w3.org:80	호스트와 포트 번호
self.location.hostname	dev.w3.org	호스트
self.location.port	80	포트 번호
self.location.pathname	/html5/worker/work.js	경로
self.location.search	?arg1=a&arg2=b	질의문
self.location.hash	#apis-available-to-workers	해시 식별자

2. 웹 소켓

2.1 웹 페이지 요청과 문제점

- 웹은 요청/응답 패러다임 기반으로 구축되어 왔다. 클라이언트가 웹 문서를 로드하고 사용자가 다음 문서를 요청하기 전까지 어떤 일도 발생하지 않는다. 정보를 얻기 위해서는 클라어언트에서 서버에 항상 요청을 해야 한다. 실시간 정보를 주고 받기 위해서는 양방향 통신이 가능해야 하고, 이를 위해 플렉스, 자바 애플릿, 실버라이트 등을 사용한다. 이것들은 순수 웹 환경이 아닌 별도의 런타임을 별도의 플러그인 형태로 설

치해야 사용 가능하다.

2.2 웹 소켓

- 순수 웹 환경에서 실시간 양방향 통신을 위한 스펙이다.
- "소켓"은 네트워크 상에서 서버와 클라이언트 두 개의 프로그램의 특정 포트를 통해 양방향 통신이 가능하도록 만들어주는 소프트웨어 장치이다.
- 요청을 한 번만 하고 웹 소켓 연결이 한 번 이루어지면, 지속적으로 연결된 TCP 라인을 통해 서버와 클라이언트 모두 언제든지 실시간으로 메시지를 송수신할 수 있도록 하는 HTML5의 새로운 사양이다.
- 연결 지향 양방향 전이중 통신이 가능하고 웹에서도 채팅, 게임, 실시간 주식 차트와 같이 실시간이 요구되는 응용 프로그램의 한층 효과적인 구현이 가능하다.

2.3 웹 소켓 생성자를 이용한 연결 설정

- 서버 연결 (공개 서버 http://www.websocket.org로 연결하는 것으로 가정)
- 웹 소켓 객체 생성

URL만 지정

```
var websocket = new WebSocket("ws://www.websocket.org");
```

URL 및 프로토콜 지정

```
var websocket = new WebSocket("ws://www.websocket.org", "myProtocol");
```

http, https ⇒ ws(일반 통신), wss(보안 통신)

XMPP, SOAP, 사용자 정의 프로토콜

2.4 웹 소켓 속성

속성	설명
url	생성자에 전달되는 URL반환
readyState	웹 소켓의 연결 상태를 반환 WebSocket.CONNECTING,WebSocket.OPEN, WebSocket.CLOSING, WebSocket.CLOSED
bufferedAmount	send() 메서드를 통해서 보내질 애플리케이션의 데이터의 바이트 수를 반환
extensions	서버에 의해서 선택된 확장을 반환
protocol	웹 소켓 생성자에서 지정한 프로토콜에서 서버가 선택한 프로토콜의 이름 반환
binaryType	메시지 송수신할 때 이진 데이터의 경우에는 "Blob", 배열버퍼 형태로 처리할 때는 "arraybuffer"로 지정

2.5 웹 소켓 이벤트

- 모든 웹 소켓 API는 이벤트 위주로 동작된다. 따라서 수신되는 데이터를 처리하고 연결 상태를 변경하기 위해서는 웹 소켓 객체를 대상으로 이벤트를 감시하면 된다.

```
onopen:  웹 소켓 연결 요청 시 서버에서 응답하여 연결이 설정되면 발생한다.
onerror : 예기치 못한 오류가 있을 때 발생한다.
onclose : 웹 소켓 연결이 끊어지면 발생한다.
onmessage : 메시지가 수신되는 순간 발생한다.
websocket.이벤트 = function(event) {
  //해당 이벤트에 대한 처리
};
```

2.6 웹 소켓 메서드

- send() 메서드를 통해서 클라이언트에서 서버로 메시지를 전송할 수 있다.

```
websocket.send("데이터");
```

```
websocket.onopen = function(event) {
    websocket.send("데이터");
}
```

open 이벤트에서 전송

```
function EventHandler(data) {
    if ( websocket.readyState == WebSocket.OPEN ) {
       websocket.send(data);
    else { //다른 작업을 처리 }
}
```

readyState 속성을 검사해서 열려있는 상태일 때만 전송

- close() 메서드는 웹 소켓의 연결을 종료시킬 수 있다.

```
websocket.close();
```

```
websocket.close(1000, "정상 종료");
```

code → 웹 소켓 연결 종료 코드

reason → 웹 소켓 연결 종료 이유

2.7 웹 소켓 클라이언트의 기본 형식

```
//서버 연결
var websocket = new WebSocket("ws://...");
// 데이터 송신
function doSend() {     websocket.send("data");}
// 데이터 수신시 호출
websocket.onmessage = function(event) {     //(서버→클라이언트) 메시지 처리}
//서버 연결시 호출
websocket.onopen = function(event) {     //이벤트 처리: alert("서버 연결 완료");}
//서버 종료시 호출
websocket.onclose = function(event) {     //이벤트 처리: alert("서버 연결 종료");}
```

예제 웹 소켓

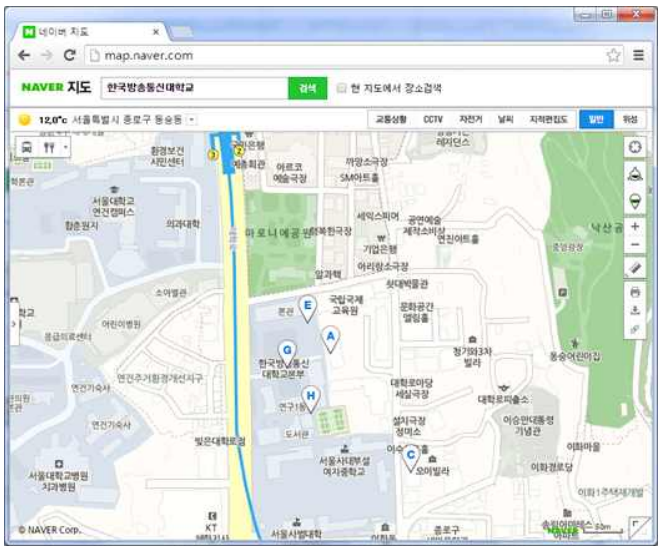
```
<!DOCTYPE html><html><head>
<script>
var wsUri = "ws://echo.websocket.org/"; var output;
function init() {
    output = document.getElementById("output");
    testWebSocket(); }
function testWebSocket() {
    websocket = new WebSocket(wsUri);  //웹 소켓 생성
    websocket.onopen = function(evt) { onOpen(evt) };  //연결되었을 때
    websocket.onclose = function(evt) { onClose(evt) };  //종료 되었을 때
    websocket.onmessage = function(evt) { onMessage(evt) };  //메시지가 수신
    websocket.onerror = function(evt) { onError(evt) }; }  // 웹 소켓 오류 발생 할 때
function onOpen(evt) {
  writeToScreen("CONNECTED");
  doSend("WebSocket rocks");  }
function onClose(evt) {
  writeToScreen("DISCONNECTED");  }
function onMessage(evt) {
    writeToScreen('<span style="color: blue;">RESPONSE: ' + evt.data +'</span>');
    websocket.close(); }
function onError(evt) {
  writeToScreen('<span style="color:red">ERROR:</span> '+ evt.data); }
function doSend(message) {
    writeToScreen("SENT: " + message);
    websocket.send(message);    }
function writeToScreen(message) {
    var pre = document.createElement("p");
    pre.style.wordWrap = "break-word";
    pre.innerHTML = message;
```

```
    output.appendChild(pre); }
 window.addEventListener("load", init, false);
</script>
</head>
<body>
    <h2>WebSocket Test</h2>
    <div id="output"></div>
</body>
</html>
```

3. 위치 정보

3.1 위치 정보

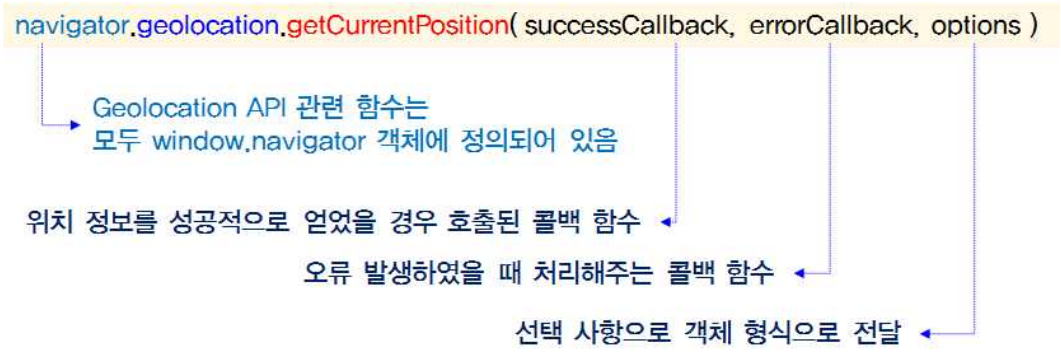
- 지도 서비스



- 지도를 이용한 현재의 위치 정보를 검색할 때 사용하는 것이 Geolocation API 이다.
- 위치 정보를 얻기 위해서는 사용자의 동의를 구해야 한다.
- 위치 정보 데이터를 얻는 방법은 IP 주소 기반, GPS 기반, WiFi 기반, 휴대전화 기반이 있다.

3.2 현재 위치 정보 얻기

- 현재의 위치 정보를 한 번만 얻기 위해서는 getCurrentPosition() 메서드를 사용한다.



3.2.1 위치 정보 관련 속성_첫 번째 인자

```
function MyPosition(position) {
document.getElementById("latitude").textContent=position.coords.latitude;   //위도
document.getElementById("longitude").textContent=position.coords.longitude; //경도
document.getElementById("accuracy").textContent=position.coords.accuracy; //위도,경도 정밀도
document.getElementById("altitude").textContent=position.coords.altitude; // GPS 고도
document.getElementById("altitudeAccuracy").textContent=position.coords.altitudeAccuracy; // GPS 고도의 정밀도
document.getElementById("heading").textContent=position.coords.heading; // 이동방향(각도)
document.getElementById("speed").textContent=position.coords.speed; // 이동속도(초당 미터)
document.getElementById("timestamp").textContent=position.timestamp; // 위치정보를 얻은 시각
}
```

3.2.2 위치 정보 오류_두 번째 인자

- 위치 정보를 반드시 얻을 수 있는 것은 아니므로 위치 오류 정보에 대한 인터페이스에서 정보를 얻을 수 있다.

```
function show_Error(error) {
 switch(error.code) {
   case error.PERMISSION_DENIED:   alert("사용 권한 오류: " + error.message); break;
   case error.POSITION_UNAVAILABLE:   alert("위치 파악 오류: " + error.message); break;
   case error.TIMEOUT:   alert("시간 초과 오류: " + error.message); break;
   default:   alert("알 수 없는 오류 발생" + error.message);
   }
}
navigator.geolocation.getCurrentPosition(MyPosition, show_Error);
```

3.2.3 위치 정보 옵션_세 번째 인자

- 위치정보를 얻는데 소요되는 시간 및 캐시의 유효기간을 지정하고자 한다면 세 번째

옵션에 정보들을 설정하면 된다.

enableHighAccuracytrue : 기본값은 false이고, 정확도가 높은 위치 정보를 얻도록 요청할 때 사용한다.
timeout :위치 정보를 얻을 때까지 기다릴 시간(제한 시간) 설정한다.
maximumAge : 캐시된 위치 정보의 유효 시간(ms) 설정한다.

예제

```
<!DOCTYPE html><html><head>
<style>table { width: 450px; }      th,td { border:1px solid black;</style>
<script>
window.onload = function() {
 var options = { enableHighAccuracy: true,  //정확한 위치 정보 요구
 timeout: 5000,                    //최대 대기 시간(밀리초)
 maximumAge: 0                        //캐시 유효 시간(밀리초)
}
if (navigator.geolocation)
 navigator.geolocation.getCurrentPosition(MyPosition, show_Error, options);
}
function MyPosition(position) {
document.getElementById("latitude").textContent = position.coords.latitude;
document.getElementById("longitude").textContent = position.coords.longitude;
document.getElementById("accuracy").textContent = position.coords.accuracy;
document.getElementById("altitude").textContent = position.coords.altitude;
document.getElementById("altitudeAccuracy").textContent = position.coords.altitudeAccuracy;
document.getElementById("heading").textContent = position.coords.heading;
document.getElementById("speed").textContent = position.coords.speed;
document.getElementById("timestamp").textContent = position.timestamp;
}
function show_Error(error) { //오류 발생시 처리할 메서드 지정
 switch(error.code) {
 case error.PERMISSION_DENIED: alert("사용 권한 오류:" + error.message ); break;
 case error.POSITION_UNAVAILABLE: alert("위치 파악 오류:" + error.message ); break;
 case error.TIMEOUT: alert("시간 초과 오류: " + error.message ); break;
    default: alert("알 수 없는 오류 발생"  + error.message ); break;
 }
}
</script>
</head>
<body>
<h2>Geolocation API - 위치 정보 나타내기</h2><hr />
<table>
```

```
<tr><th>상세한 내용</th><th>위치 정보</th></tr>
<tr><td>위도 </td><td id="latitude">.</td></tr>
<tr><td>경도 </td><td id="longitude">.</td></tr>
<tr><td>위도 및 경도의 정밀도</td><td id="accuracy">.</td></tr>
<tr><td>GPS 고도</td><td id="altitude">.</td></tr>
<tr><td>GPS 고도의 정밀도</td><td id="altitudeAccuracy">.</td></tr>
<tr><td>이동 방향</td><td id="heading">.</td></tr>
<tr><td>이동 속도</td><td id="speed">.</td></tr>
<tr><td>위치 정보를 획득한 시간</td><td id="timestamp">.</td></tr>
</table>
</body>
</html>
```

3.3 연속적인 위치 정보 얻기

- 현재의 연속적인 위치 정보를 얻기 위해서는 watchPosition() 메서드를 사용한다.
- clearWatch()메서드를 호출하기 전까지는 계속해서 위치 정보를 감시한다. clearWatch() 메서드에서는 watchPosition() 메서드 호출시에 반환되는 식별 ID를 인수로 지정한다.

```
var 식별ID=navigator.geolocation.watchPosition(successCallback, errorCallback, options);
navigator.geolocation.clearWatch(식별ID);
```

3.4 구글 지도 활용하기

- Google Maps Javascript API v3는 공식적으로 무료인 자바스크립트 API이다.
  https://developers.google.com/maps/documentation/javascript/tutorial?h1=ko
- 지도 서비스 관련 API 제공 사이트
  네이버 지도 → http://developer.naver.com/wiki/pages/MapAPI
  다음 지도 API 3 → http://dna.daum.net/apis/maps

예제 지도 서비스 활용

```
<!DOCTYPE html> <html>  <head>
 <style>
      html, body, #map-canvas { height: 100%; margin: 0px; padding: 0px }
</style>
 <meta name="viewport" content="initial-scale=1.0, user-scalable=no">
 <script src="https://maps.googleapis.com/maps/api/js?v=3.exp"></script>
 <script>
var map, lat = 37.5010226, lng = 127.0396037;
function initialize() {
```

```
 var mapOptions = { zoom: 12, //지도의 확대 비율을 지정한다.
 center: new google.maps.LatLng(lat, lng), //현재의 위치 정보를 기준으로 한다.
 mapTypeId: google.maps.MapTypeId.ROADMAP
};
//현재 지도를 중심으로 구글 지도를 표시하도록 한다.
map = new google.maps.Map(document.getElementById('map-canvas'), mapOptions);
}
google.maps.event.addDomListener(window, 'load', initialize);
 </script>
 </head>
<body>
  <div id="map-canvas" style="width:500px;height:500px"></div>
</body>
</html>
```