# 시맨틱 Tag

# 장점

- ✓ 접근성이 좋아짐
- ✓ SEO(Search Engine Optimization)
- ✓ 수정이 용이
- ✓ 코드 가독성이 좋아짐

# nav

- ✓ 네비게이션을  표현하기 위한 태그
- ✓ 페이지 안의 모든 링크의 그룹이 nav 요소로 기술될 필요는 없음
- ✓ 사이트내의 메뉴 등
- ✓ nav 요소로는 그 페이지의 주요 링크만 처리

# header

- ✓ 소개나 네비게이션 기능들의 묶음을 나타냄
- ✓ header 요소는 목차나 검색창, 로고 등을 포함할 수 있음

# footer

- ✓ 저자나 저작권 등을 포함할 수 있음

# hgroup

- ✓ 제목과 그와 관련된 부제목을 묶는 역활
- ✓ 페이지 전체 구조에 대한 개념으로 쉽게 눈에 들어오게 하는 역활

# section

✓ 웹콘텐츠들을 그룹으로 묶어주는 역할

# article

- ✓ 문서내의 독립적인 하나의 콘텐츠를 구분
- ✓ 블로그 글이나 뉴스본문
- ✓ 주로 section 태그와 같이 사용

# aside

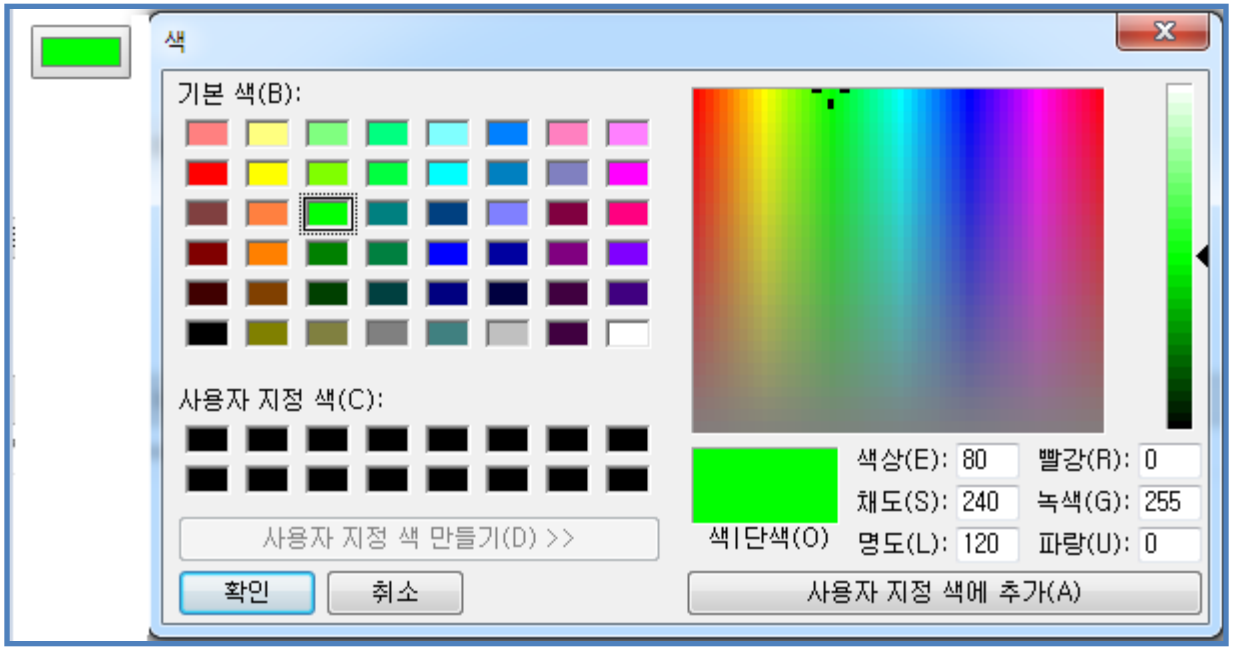
- ✓ 문서의 주 내용이 아닌 관련성이 낮은 내용들은 aside로 표시
- ✓ 본문과 직접적으로 상관이 없는 관련 사이트 링크나 광고
- ✓ 블로그 등에 있는 사이드 바와 같은 형태

# Input Tag

# Input tag - color

✓ color

- 기본 색상이 화면에 보여짐
- 색상을 선택 시 색상 선택 창이 보여짐
- `<input type="color" name="co" value="#00ff00" />`

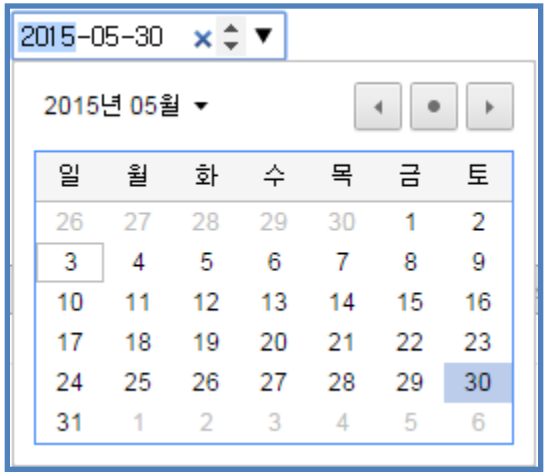


# Input tag - date

✓ date

- 날짜를 선택하는 창
- min, max : 날짜 선택의 폭을 정함

```
<input type="date" name="da" value="2015-05-30" />
```



# Input tag – email, month, week

- ✓ email
  - submit 시 이메일 창의 값 체크

```
<input type="email" name="email" />
```

- ✓ month
  - 년, 월을 선택하는 창

- ✓ week
  - 년, 주를 선택하는 창

# Input tag - number

- ✓ number
  - 숫자를 선택하는 입력 창
  - min , max : 숫자 선택의 범위를 정함
  - step : 값의 증가수치를 정함

```
<input type="number" name="number"
        min="0" max="100" step="10" />
```

# Input tag - range

- ✓ range
  - 슬라이더바 형태의 선택 창
  - min , max : 슬라이더 선택의 범위를 정함
  - step : 값의 증가수치를 정함

```
<input type="range" name="range"
        min="0" max="100" step="2" value="0“ />
```

# Input attribute

- ✓ max
- ✓ min
- ✓ pattern
- ✓ required
- ✓ step

# Canvas

# <canvas> 태그

**<canvas>  </canvas>**

- 그래픽을 위한 컨테이너 역할 **(** 선, 사각형, 원, 텍스트, 이미지 **)**
- 실제 그래픽을 그리기 위해서는 스크립트를 사용

# <canvas> 태그의 속성

**<canvas id="myCanvas" width="500" height="300"> </canvas>**

- **id :** 스크립트에서 접근을 하기 위해 사용
- **width**
- **height**

# Canvas 태그 지원

**<canvas id="myCanvas" width="500" height="300">**

브라우져가 캔버스를 지원하지 않습니다.

**</canvas>**

- 브라우져가 **HTML5 Canvas** 태그를 지원하지 않으면 화면에 보이지 않는다.
- 브라우져가 지원하지 않을 경우 화면에 보여질 내용을 태그 사이에 명시한다.

# Javascript 캔버스 사용

```
<canvas id="myCanvas" width="500" height="300">
   브라우져가 캔버스를 지원하지 않습니다.
</canvas>
```

---

```
var canvas = document.getElementById('myCanvas');
if (canvas.getContext) {
  var ctx = canvas.getContext('2d');
  // 그리기 관련 코드 작성
}
```

# 사각형 그리기 예제작성

- ✓ strokeRect(x, y, w, h) : 사각형 윤곽 그리기
- ✓ fillRect(x, y, w, h) : 사각형 색으로 채우기
- ✓ clearRect(x, y, w, h) : 지정된 사각형 영역 지우기

# 색 변경하기 예제작성

- ✓ 색을 변경한 후 그린다.
- ✓ `ctx.fillStyle = "#E01B6A";`
- ✓ `ctx.fillRect(0, 0, 100, 100);`

# 선 그리기 예제작성

- ✓ beginPath ( ) : 선 그리기 설정, 패스지정 초기화
- ✓ closePath ( ) : 현재까지 지정한 패스를 종료
- ✓ moveTo (x, y) : 선 그리기 시작 좌표로 이동
- ✓ lineTo (x, y) : 좌표를 이동하고 이전좌표에 선 긋기
- ✓ stroke() : 두 좌표 사이에 선 그리기
- ✓ lineWidth : 선 넓이
- ✓ strokeStyle : 선 색

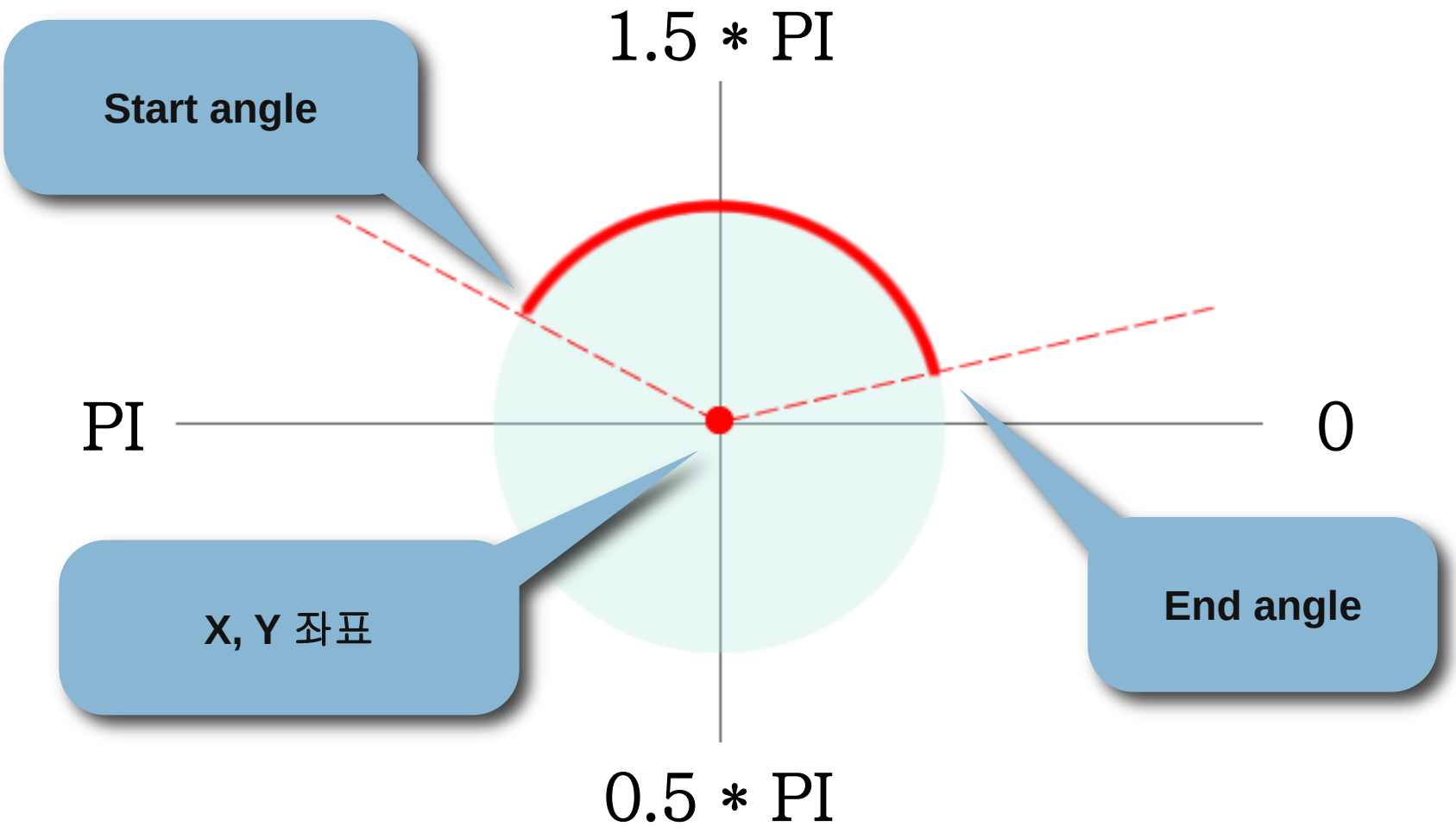
# 선 그리기 예제작성

- ✓ lineCap : 선의 끝 모양 지정,
  - butt : 정확한 좌표점 부터 시작
  - round, square : 좌표 밖에 표시
- ✓ lineJoin : 선이 만나는 지점의 모양
  - bevel(모서리가 깍인 모양), round, miter
- ✓ fill : 선을 연결하고 선안을 채우기

# 원 그리기 예제작성

- ✓ arc (x, y, r, st, ed, 방향(true, false))
- ✓ x, y : 중심좌표
- ✓ st, ed : 시작각과 끝각, 라디안으로 계산
  - 라디안 = 각도 * Math.PI / 180
- ✓ 방향 : true (반시계), false (시계)

# 원 그리기



# 텍스트 그리기 예제작성

- ✓ **fillText (text, x, y[, maxWidth]) :** 채워진 텍스트
- ✓ **strokeText(text, x, y[, maxWidth]) :** 외곽선만 표시
- ✓ **font :** 글꼴, 굵기, 글꼴 크기 지정
- ✓ **textAlign :** 문자열 가로방향 위치

  **left, right, center, start, end**
- ✓ **textBaseline :** 텍스트가 표시되는 기준선

  **top, middle, bottom, alphabetic**(아래)

# 이미지 그리기 예제작성

- ✓ **drawImage(img, x, y)**
  - ➢ 이미지를 그릴 **X, Y** 좌표
- ✓ **drawImage(img, x, y, w, h)**
  - ➢ 이미지를 그릴 **X, Y** 좌표 및 넓이와 높이
- ✓ **drawImage(img, sx, sy, sw, sh, x, y, w, h)**
- ✓ **globalAlpha : 투명도**

# Audio

# 태그

- ✓ <audio>…</audio>
- ✓ <source src=“ “ type=“ “ />
- ✓ 비디오와 오디오와 같은 미디어 엘리먼트를 위한 리소스를 정의

```
<audio controls>
  <source src="My Type.mp3" type="audio/mpeg">
  오디오 태그를 지원하지 않는 브라우져 입니다.
</audio>
```

# MIME Types

✓ Audio 사용 포멧

| Format | MIME-type |
|---|---|
| MP3 | audio/mpeg |
| Ogg | audio/ogg |
| Wav | audio/wav |

✓ 브라우져 지원 여부

| Browser | MP3 | Wav | Ogg |
|---|---|---|---|
| Internet Explorer | YES | NO | NO |
| Chrome | YES | YES | YES |
| Firefox | YES | YES | YES |
| Safari | YES | YES | NO |
| Opera | YES | YES | YES |

# 이벤트 및 속성

- ✓ **playing**
  - ➢ 플레이  시 발생되는 이벤트
- ✓ **pause**
  - ➢ 일시중지 시 발생되는 이벤트
- ✓ **currentTime**
  - ➢ 현재 플레이되고 있는 시간
- ✓ **duration**
  - ➢ 플레이 시간
- ✓ **timeupdate**
  - ➢ 플레이시간이 바뀔때 호출되는 이벤트

# Video

# 개요

- ✓ <video>…</ video>
- ✓ <source  src=“ “ type=“ video/ogg“ />
- ✓ Plug-in 없이 비디오의 재생이 가능해 짐
- ✓ 브라우저별로 재생 할 수 있는 미디어 형식이 다름

```
<video width="400" controls>
  <source src="test.mp4" type="video/mp4">
  <source src="test.ogv" type="video/ogg">
  <source src="test.webm" type="video/webm">
  브라우져가 video 태그를 지원하지 않습니다.
</video>
```

# 속성

| 속성명 | 설명 |
| --- | --- |
| controls | 재생 Control 표시를 제어 |
| autoplay | 동영상 로딩 시 자동 재생 |
| poster | 동영상이 다운로드 중이거나 버퍼링 중에 나타낼 이미지를 지정 |
| loop | 동영상을 반복 재생 |
| width | 동영상의 너비 지정 |
| height | 동영상의 높이 지정 |

# 제어 함수

| 속성명 | 설명 |
| --- | --- |
| load() | 동영상을 다시 읽어들임(재생하지 않음) |
| play() | 동영상 재생 |
| pause() | 재생 중인 동영상을 일시 중지 |
| canPlayType(type) | MIME 타입에 따라 재생 가능한 video 요소인지 여부 테스트 |

# 쓰기 가능한 속성

| 속성명 | 설명 |
| --- | --- |
| autoplay | 미디어 클립이 자동 재생되도록 설정 |
| muted | 음소거 인지 반환, 오디오 음소거 수행 및 취소 |
| volume | 음량을 표시. 0.0 ~ 1.0 까지 값을 가짐 |
| loop | 클립이 재생 완료 후 다시 재생되면 true를 반환 하거나 클립이 순환 재생되도록 설정 |
| controls | 사용자 컨트롤을 보여주거나 숨긴다. 현재 컨트롤이 보여지는지 여부도 알 수 있음 |

# Drag&Drop

# 개요

- ✓ 사용하려는 엘리먼트에 속성을 설정
- ✓ 관련 이벤트

1. dragstart : 드레그 시작시 발생
2. dragend : 드레그 종료시 발생
3. drag : 드레그 중일때
4. dragenter : 드랍존으로 드레그가 들어왔을때
5. dragleave : 드랍존에서 드레그가 나갔을때
6. dragover : 드랍존에 있는 상태
7. drop : 드랍존에서 드레그가 종료되었을때

# 속성

✓ 엘리먼트에 draggable 속성을 설정

draggable 설정 가능한 값

- true, auto, false

```
<img width="300" height="180" draggable="true"
     src="../images/pic1.jpg" />
```

# 이벤트

```
<img id="img1" width="300" height="180" draggable="true"
     src="../../images/pic1.jpg"
     ondragstart="console.log('드레그 시작')"
     ondrag="console.log('드레그 중....');" />

<div id="dropzone"
     ondragenter = "console.log('드랍존 들어옴');"
     ondragover  = "console.log('드랍존에 있는 상태');"
     ondragleave = "console.log('드랍존을 벗어남');"
 ></div>
```

# Geolocation

# Geolocation 개요

- ✓ 브라우져가 실행될 수 있는 모든 환경(데스크탑, 스마트폰, 태블릿 등) 에서 사용이 가능
- ✓ navigator.geolocation
  - 지오로케이션 API를 포함하는 객체
- ✓ 개인 정보 보호를 위해 사용 전 사용자에게 확인

# Geolocation 위치결정

- ✓ IP 기반 : 물리적인 위치 정보 활용
- ✓ 와이파이 : 위치를 삼각 측량
- ✓ 휴대폰 : 삼각측량법 사용, 한 곳 이상의 기지국에서 떨어진 거리를 계산 ( 기지국이 많을 수록 위치 정보가 정확)
- ✓ GPS (Global Positioning System)

# Geolocation 사용 여부

```
if (navigator.geolocation) {
    alert("geolocation을 지원하는 브라우저입니다.");
} else {
    alert("geolocation을 지원하지 않는 브라우저입니다.");
}
```

# Geolocation 제공 함수

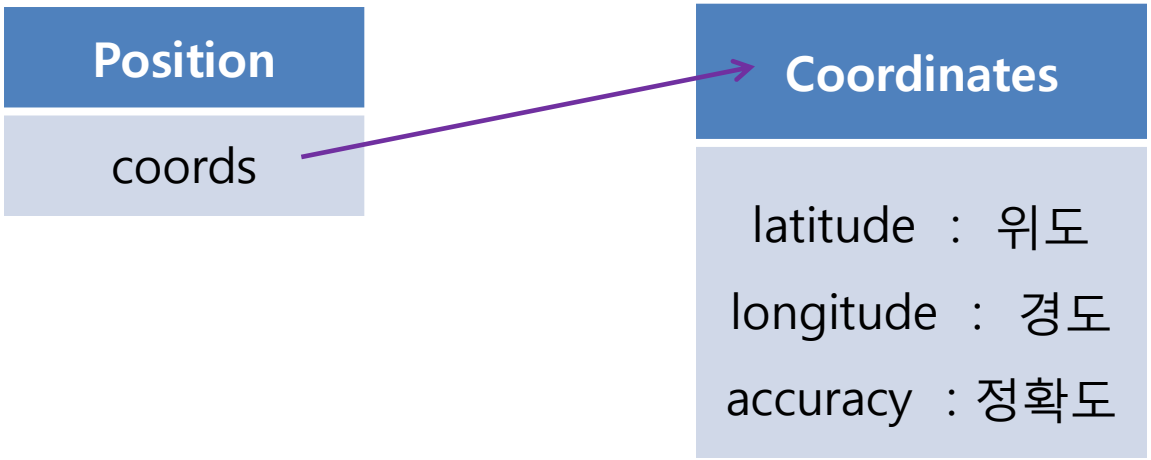
- ✓ getCurrentLocation ( successHandler,  errorHandler,  options )
- ✓ watchId = watchPosition ( successFn, errorFn, options )
- ✓ clearWatch ( watchId )

# 현재 위치 정보 가져오기

- ✓ getCurrentLocation ( successHandler,  errorHandler,  options )
- ✓ 매개변수
  - successHandler : 필수

```
function successHandler ( position ) {
    // 위치정보를 가지고 있는 coordinates 객체 반환
    var coords = position.coords;
    console.log (“위도 : “ + coords.latitude );
    console.log (“경도 : “ + coords.longitude );
}
```

# 현재 위치 정보 가져오기


Position
coords
Coordinates
latitude : 위도
longitude : 경도
accuracy : 정확도

# 현재 위치 정보 가져오기

- errorHandler : 선택

  1) error.code

     0 : 알려지지 않은 에러

     1 : 권한 거부

     2 : 위치 정보를 찾지 못함

     3 : 요청 응답 시간 초과

  2) error.message : error.code 가 0번 또는 2번일 경우

```
function errorHandler ( error ) {
    console.log ( error.code );
}
```

# 현재 위치 정보 가져오기

- options : 선택, 작동 방식을 변경할 수 있는 옵션 설정부분

  1) enableHighAccuracy : 정확도를 높게 조정하는 속성

  2) timeout : Infinity (기본값)

  3) maximumAge : 위치 계산할 최대 주기를 지정할 때 사용,

  0(기본값), 항상 위치를 재계산

# 현재 위치 정보 가져오기

옵션 예 >

{ maximumAge:  600000}

브라우저가 10분이 넘지 않은 위치 정보를 갖고 있다면 그 위치를 사용할 것임, 그렇지 않다면 새로운 위치가 필요함

{ timeout: 1000,  maximumAge:  600000}

나는 10분이 넘지 않는 위치 정보만 원함, 10분이 넘은 위치 정보라면 새로운 위치를 요구할 것임.  1초내에 새 위치를 가져 올 수 있음

# 현재 위치 정보 가져오기

옵션 예 >

{ timeout:0,  maximumAge: Infinity }

나는 이미 가져온 위치만 원합니다. 이전에 가져온 위치가 없다면 에러 핸들러를 호출할 것임  새로운 위치는 필요없음. 나는 오프라인용임

{ timeout: Infinity,  maximumAge:  0 }

나는 새로운 위치만 원합니다. 브라우저가 필요할 때 언제든 위치를 가져올 수 있습니다.

# 위치 정보 업데이트

- ✓ watchId = watchPosition ( successFn, errorFn, options )

  현재 위치 갱신하기

  위치 정보가 변경(장치가 움직였거나 좀 더 정확한 위치 정보가 도착했을때) 되었다면 갱신된 위치 정보로 호출되는 콜백 함수

- ✓ clearWatch ( watchId )

  사용자의 위치 정보를 갱신할 필요가 없을 경우 호출

# 활용

- ✓ 근처 친구 찾기
- ✓ 이동하는 경로를 기록으로 남길 때
- ✓ ….

# localstorage

# Local Storage 개요

- ✓ 기존 쿠키의 단점
  1. 용량이 제한(4k)
  2. 매번 쿠키를 주고 받음
  3. 바이러스나 악성코드 등 악용될 가능성이 있음

- ✓ 클라이언트의 디스크에 데이터를 저장
- ✓ 크기에 제한이 없음(도메인당 5MB)
- ✓ 서버로 보내지지 않음
- ✓ 유효 기간의 제한이 없음
- ✓ Javascript 객체를 저장할 수 있음
- ✓ 브라우져별 관리

# Local Storage 지원 여부

```
if ( window.localStorage ) {
    console.log( “localStorage를 지원하지 브라우져“);
} else {
    console.log( “localStorage를 지원하지 않는 브라우져“);
}
```

# 동작 방식 이해

- ✓ 웹페이지를 통해 하나 이상의 키/값 쌍을 브라우저 저장소에 저장
- ✓ 이후 키를 사용해서 값을 추출
- ✓ 연관배열 처럼 사용이 가능

저장 : localStorage.setItem(“memo”, “오늘은 자바 공부”);

추출 : var memo = localStorage.getItem(“memo”);

localStorage[“memo”] = “오늘은 자바 공부”;

var memo = localStorage[“memo”] ;

# 속성 및 메서드

- ✓ length : 저장된 키/값 집합의 크기
- ✓ key(index) : 해당 위치의 키를 추출
- ✓ setItem(key, value) : 스토리지에 저장
- ✓ getItem (key) ： 값을 추출
- ✓ removeItem (key) : 값을 삭제
- ✓ clear ( ) : 스트리지의 모든 값을 지운다.

# 이벤트

- ✓ key : 변경된 key
- ✓ oldValue : 변경되기 전의 값, 새로운 키 등록이라면 null
- ✓ newValue : 변경된 후의 값, 값이 삭제 되었을 때는 null