MobileApp 과정 #2

2019.4

박재성

목차

실습 설정: Deprecating Powerful Features on Insecure Origins

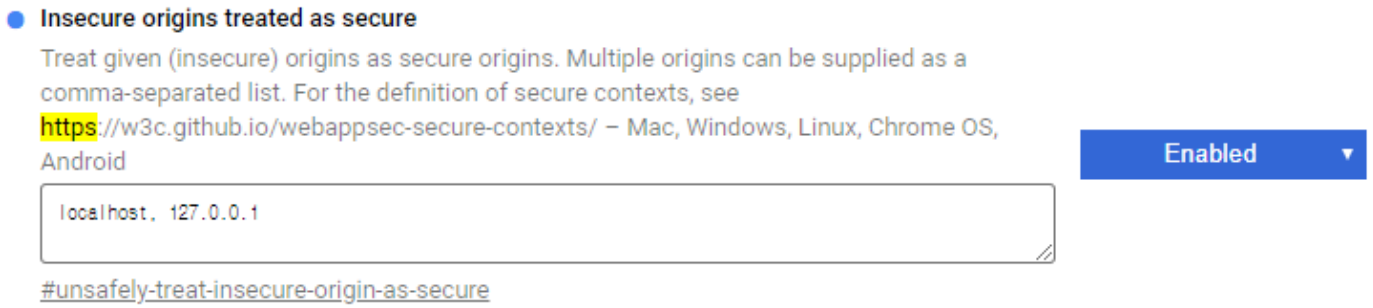
아래의 기능들에 대해선 보안상 'https' 프로토콜이 아닌 경우, 사용이 제한된다.

- Geolocation — requires secure origins as of M50
- Device motion / orientation
- EME
- getUserMedia
- AppCache
- Notifications

chrome://flags/#unsafely-treat-insecure-origin-as-secure

테스트를 위해, Chrome 주소 창에 다음 값을 입력 후,

- (1) 활성화 처리
- (2) 입력 창에 127.0.0.1과 localhost를 입력



1. Geolocation

GeoLocation?

개요

- 웹사이트 방문자의 위치 정보인 위도(latitude)와 경도(longitude)에 대한 정보를 얻을 수 있으며, 이 정보를 통해 위치기반(LBS - Location Based Service) 서비스를 할 수 있다.

- 위치에 대한 정보는 GPS, 사용자 IP 주소, RFID, WiFi, 블루투스, 기기의 MAC Address 그리고 GSM/CDMA 셀 ID 정보 등을 활용해서 얻지만, 위치를 정확하게 반환하는 것을 보장하지는 않는다.

- GeoLocatoin은 옵트(Opt-In)인 방식으로, 사용자가 동의하는 경우에만 위치 정보를 전송하게 된다.

GeoLocation 사용하기

지원여부 확인

브라우저에서 지원되는지 확인하기

```
if(navigator.geolocation) {
        …
}
```

→ Spec 문서 : http://www.w3.org/TR/geolocation-API/

GeoLocation 사용하기

위치 값 얻기 - getCurrentPosition()

현재 위치 정보 얻기

```
navigator.geolocation.getCurrentPosition(
        successCallback,        → 성공 콜백
        errCallback,            → 오류 발생시 (optional)
        options                 → 추가 옵션 (optional)
);
```

GeoLocation 사용하기

위치 값 얻기 - getCurrentPosition()

현재 위치 정보 얻기

```
navigator.geolocation.getCurrentPosition(function(position) {
        position.coords.latitude;       // 위도
        position.coords.longitude;      // 경도
        position.coords.altitude;       // 고도
});
```

GeoLocation 사용하기

지속적인 위치 값 얻기 - watchPosition()

watchPosition은 위치 값을 지속적으로 얻을 때 사용 (예. 네비게이션)

```
const watchId = navigator.geolocation.watchPosition(
        successCallback,        → 성공 콜백
        errCallback,            → 오류 발생시 (optional)
        options                 → 추가 옵션 (optional)
);
```

삭제할 때는 아래와 같이 한다.

```
navigator.geolocation.clearWatch(watchId);
```

GeoLocation 사용하기

Position 객체

Position	
coords	실제 위치 정보에 대한 값을 담고 있다.
timestamp	위치 정보를 얻은 시간(DOMTimeStamp)을 의미한다.

Coords 객체

Coordinates	Type	
latitude	double	위도 값
longitude	double	경도 값
altitude	double	고도 값
accuracy	double	위도와 경도의 정밀도 레벨. 미터로 표현
altitudeAccuracy	double	고도의 정밀도 레벨. 미터로 표현.
heading	double	도(degree)로 표현되며, 시계방향 북쪽으로부터 시작.
speed	double	기기의 속도로 초당 미터의 값으로 표현

GeoLocation 사용하기

Option에 사용할 수 있는 값

PositionOptions	Type	
enableHighAccuracy	boolean	보다 정확한 결과를 얻고자 할 때 설정할 수 있다. 하지만 설정되는 경우 응답 속도가 느려질 수 있는데, 이 속성은 보다 정확한 정보를 필요로 하지 않는 경우 사용될 수 있도록 한 것이다. 즉, 단순하게 geolocation provider 정보를 사용하지 않고 보다 정확한 위치 정보를 얻기 위해 GPS 등을 이용하는 경우 모바일 기기들에서는 배터리 소모가 빨라지기 때문에, 모바일 기기들에게는 이 속성을 사용하는 것이 적합하지 않을 수도 있기 때문이다.
timeout	long	위치정보 호출 후, 최대 응답대기 시간을 의미한다. 단위는 milliseconds(ms) 이다.
maximumAge	long	지정된 시간보다 크지 않은 경우, 캐싱된 위치정보를 사용하도록 한다. 단위는 milliseconds(ms) 이다.

GeoLocation 사용하기

Error 객체

PositionError	
code	에러코드 : PERMISSION_DENIED = 1 POSITION_UNAVAILABLE = 2 TIMEOUT = 3
message	에러 메시지

GeoLocation 사용하기

모든 옵션을 적용한 경우

```
navigator.geolocation.getCurrentPosition(
        pos => { // 성공시
                alert(pos.coords.latitude);
                alert(pos.coords.longitude);
        },
        err => {  // 실패시
                alert(err.message);
        },
        {       // 옵션적용
                enableHighAccuracy : false,
                timeout : 2500,
                maximumAge : 75000
        }
);
```

실습

- 위치 정보를 얻는 간단한 코드를 작성해 봅니다.

File : 1.gelocation/1.html, 2.html

2. Network Information

Network Connection?

개요

- 사용자의 네트워크 상황을 확인 할 수 있음.
- 일반적으로 크기가 큰 자원(동영상…)을 실행시키기 전에 확인하여 UX을 개선할 수 있음.
- 대다수 모던 브라우저에서 사용가능

→ https://caniuse.com/#feat=netinfo

Network Connection 사용하기

온/오프라인 확인 속성

Boolean 값을 갖는 속성을 통해 온/오프라인 구분

```
if(navigator.onLine) {
    // 온라인 상태
} else {
    // 오프라인 상태
}
```

→ https://developer.mozilla.org/en-US/docs/Web/API/NavigatorOnLine/onLine

Network Connection 사용하기

온/오프라인 이벤트

"online" 또는 "offline" 이벤트를 window에 바인딩

```
// 온라인 일때 발생
window.addEventListener("online", function(e) {
    ...
});

// 오프라인 일때 발생
window.addEventListener("offline", function(e) {
    ...
});
```

Network Information 사용하기

지원여부 확인

브라우저에서 지원되는지 확인하기

```
if(navigator.connection) {
        ...
}
```

Android 환경과 Chrome에서만 사용가능하며, 크롬 버전에 따라 사용할 수 있는 속성이 다를 수 있다. (ex. Samsung Internet과 Chrome)

→ https://caniuse.com/#feat=netinfo

→ Spec 문서 : http://wicg.github.io/netinfo/

Network Information 사용하기

connection

connection 객체 값

```
const conn = navigator.connection;
conn.type;
```

속성	설명
type	현재 상태 속성에 맞는 값 → bluetooth, cellular, ethernet, none, mixed, other, unknown, wifi, wimax → http://wicg.github.io/netinfo/#connection-types
effectiveType	현재 연결상태를 표현: 'slow-2g', '2g', '3g', 또는 '4g' 중 한가지 값
rtf	round-trip time rounded to the nearest multiple of 25 milliseconds
downlink	Bandwidth MB/sec
downlinkMax	최대 다운로드 MB/sec
saveData	데이터 절감 모드 설정 여부 (Boolean)

Network Information 사용하기

이벤트

```
// 연결이 변경되면 이벤트 발생
navigator.connection.addEventListener("change", function(e) {
    e.target;
    /* {
        downlink: 10
        effectiveType: "4g"
        onchange: null
        rtt: 100
        saveData: false
    } */
});
```

실습

- Network Information 실습

3. Battery Status API

Battery Status API

현재 디바이스의 배터리 정보를 얻어올 수 있다.

- Chrome과 Android에서만 사용가능
- Firefox는 보안 이슈로 인해 Firefox v52에서 제거
- Promise를 반환

```
let batteryIsCharging = false;

navigator.getBattery().then(function(battery) {
    batteryIsCharging = battery.charging;  // 충전 중인지 여부

    // 충전 상태 변경 이벤트 바인딩
    battery.addEventListener("chargingchange", function() {
        batteryIsCharging = battery.charging;
    });
});
```

→ https://caniuse.com/#search=battery
Spec 문서: https://w3c.github.io/battery/#the-navigator-interface

Battery Status API

Battery Manager는 다음의 인터페이스를 갖는다.

속성	타입	설명
charging	Boolean	배터리가 현재 충전 중인지 여부
chargingTime	Number	배터리가 완전히 충전되기 까지 남은 시간(초) → 0이면 충전이 완료된 상황
dischargingTime	Number	배터리가 완전히 방전되고 시스템이 중지 될 때까지의 남은 시간(초)
level	Float	배터리 충전 상태 레벨 값. → 0.0 ~ 1.0 사이의 값으로 표현

이벤트	설명
onchargingchange	배터리 충전 상태가 변경될 때 발생
onchargingtimechange	배터리 충전 시간이 변경될 때 발생
ondischargingtimechange	배터리 방전 시간이 변경될 때 발생
onlevelchange	배터리 수준 상태가 변경될 때 발생

실습

- Battery Status API 실습

4. Media Query

Media Query?

개요

- 다양한 크기의 기기에서 viewport 해상도에 따라 CSS을 적용하는 기술
- 예를 들어, 데스크탑과 모바일이 하나의 마크업으로 다른 모습으로 보이게 하고 싶을 때 사용할 수 있다.

→ https://developer.mozilla.org/en-US/docs/Web/Guide/CSS/Media_queries

Media Query 사용하기

기본 문법

- only : 미디어 쿼리를 지원하면 해석하라는 의미이다.
- not을 넣으면 뒤에 조건은 부정형을 의미한다.
- all : 대상 미디어를 지정. (all, screen, print, tv...)
- and : 조건문의 AND 연산 → OR 연산은 ,(콤마)로 함
- (조건문) : 해당 조건문에 맞으면 스타일이 적용

```
only all and (조건문)
```

Media Query 사용하기

CSS파일로 분리하는 방법

```
<link rel="stylesheet" type="text/css" media="all and (조건문A)"
        href="desktop.css">

<link rel="stylesheet" type="text/css" media="all and (조건문B)"
        href="mobile.css">
```

Media Query 사용하기

스타일을 분리하는 방법

```
@media all and (조건문) {
        //스타일
}
```

Media Query 조건문

Height/Width

가장 많이 사용하는 속성으로 뷰 포트의 너비와 높이를 확인하여 적용

```
@media all and (min-width:360px) and (max-width:1024px) { … }
// 뷰포트 너비가 최소 360px에서 최대 1024px 사이에 있는 경우 적용

@media all and (width:360px), (width:1024px) { … }
// 뷰포트 너비가 360px이거나 1024px이면 적용
```

Media Query 조건문

orientation
가로 모드인지 세로모드인지 확인하여 적용

```
@media all and (orientation:portrait) { … }        // 세로 모드

@media all and (orientation:landscape) { … }     // 가로 모드
```

Media Query 조건문

aspect-ratio
뷰 포트의 너비와 높이에 대한 비율을 확인하여 적용

```
@media all and (aspect-ratio:1/2)
// 뷰포트의 너비/높이가 1/2이면 적용

@media all and (min-aspect-ratio:1/2)
// 뷰포트의 너비/높이가 최소 1/2이상이면 적용
```

Media Query 조건문

resolution

출력 디바이스의 해상도(화면밀도 - pixel density)

해상도의 단위:
- dots per inch(dpi)
- dots percentimiter(dpcm)
- **dots per pixel(dppx) → Ratio pixel**

```
@media (resolution: 150dpi) { ... }  /* 150dpi인 경우 적용 */
@media (min-resolution: 2dppx) { ... }. /* 최소 pixel ratio가 2인 경우 */
@media (max-resolution: 300dpi) { ... }  /* 최대 300dpi인 경우 */
```

Media Query 조건문

기타

- device-width / device-height
- device-aspect-ratio
- color
- color-index
- monochrome
- scan
- grid

Media Query 유의사항

- 모든 기기에서 미디어 쿼리를 이용하여 적용하기는 힘들다.
- UX/기획하기가 힘들고 성능이 나빠질 수 있다.
- 적절하게 나누는 게 필요.
  - 모바일의 서로 다른 해상도는 괜찮음.
  - 데스크탑과 모바일은 고민해 봐야함.

5. Vibration API

Vibration API

모바일 기기에 vibration 기능이 있는 경우, 사용자에게 진동을 통한 알림을 줄수 있게 한다.

API는 간단하며, 진동이 진행될 시간을 ms로 지정하면 된다.

```
// 진동
window.navigator.vibrate(number);

// 진동 패턴 : [ 진동시간 ms → 정지시간 ms → 진동시간 ms, … ]
window.navigator.vibrate(Array);
```

Vibration API: 예제

사용자의 명시적인 이벤트(click 등)를 통해 호출된 경우에만 동작될 수 있으며, 이는 과도한(또는 악의적인) vibration 사용을 방지하기 위함이다.

```
<button onclick="vibrate()">진동!</button>

function vibrate() {
    navigator.vibrate(200);
}
```

```
// 200ms 동안 진동한다.
window.navigator.vibrate(200);

// 200ms 진동 → 100ms 일시정지 → 200ms 진동
window.navigator.vibrate([200, 100, 200, … ]);

// 진행중인 vibration 패턴이 있다면 취소한다.
window.navigator.vibrate(0)
```

6. Animation

Animation을 만드는 방법의 종류

애니메이션을 만드는 방법은 다음이 있다.

- setTimeout, setInterval : 타이머를 이용하는 방법
- CSS3 : transform, transition, animation을 이용하는 방법
- requestAnimationFrame : RAF을 이용하는 방법

- **WebAnimation API 사용**
  → IE/Edge 미 지원
  → iOS 미 지원 (Safari Technology Preview에서 지원 추가로 향후 iOS 지원가능성 높음)

Timer

- 주기적으로 반복하여 애니메이션 함.

```
//현재 위치에서 왼쪽을 기준으로 100px이동함.
const ele = document.querySelector("#some");
const id = setInterval(() => {
        const left = parseInt(ele.style.left) || 0;

        ele.style.left = (left + 5) + "px";

        if(left == 100) {
            clearInterval(id);
        }
},100);
```

- 매번 DOM을 변경하기 때문에 reflow가 발생.

CSS3

- 스타일로 애니메이션 할수 있음.

- transform : 오브젝트를 변경함.
  - skew : 기울기 (35deg)
  - scale : 비율 (2,2)
  - rotate : 회전 (45deg)
  - translate : 이동 (100px,20px)

- transition : 오브젝트를 속도를 지정한다.
  - 변경할 속성 : transform, width
  - 시간 : 3s
  - 속도 그래프 : ease-in-out
  - 지연 시간 : 3s

CSS3

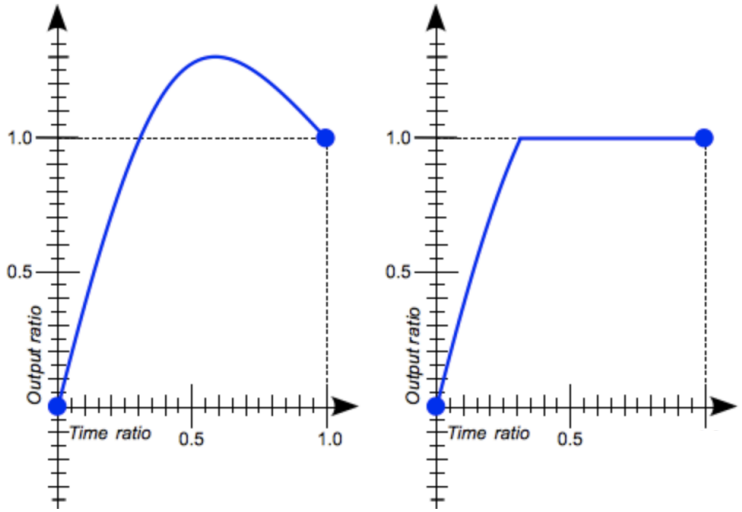
```
<style>
    #some.move {
        transform : translate3d(100px,0,0);
        transition: transform 2s ease-in-out;
    }
</style>
<script>
    const ele = document.querySelector("#some");

    ele.classList.add("move");
    ele.addEventListener("transitionend",function() {
        alert("end");
    });
</script>
```

CSS3: Transition / @keyframes

- CSS로 움직이기 때문에 실제로 DOM이 움직이는게 아니라 레이어만 이동하여 한번의 reflow 발생한다.
- 3D옵션을 사용하면 GPU에서 실행되기 때문에 보다 효율적. 레이어가 하나 더 생성되어 GPU는 레이어별로 처리.
  - 무조건 좋은건 아님. 모바일에서는 VRAM이 제약적이라 더 느릴 수 있음.
- 고정된 애니메이션이 아니면 프로그래밍하기가 힘들다.

CSS3: Timing Function

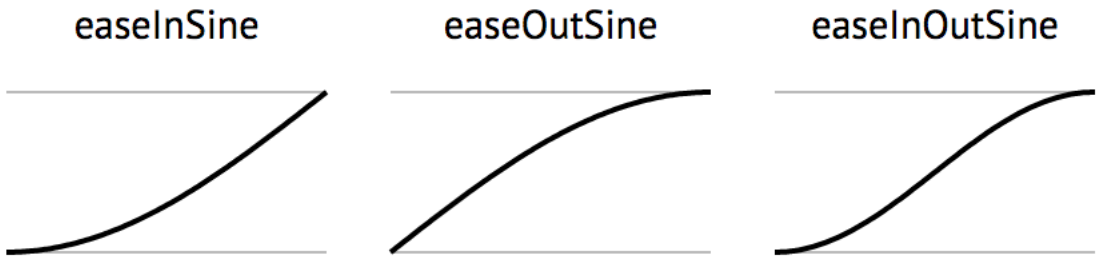
- CSS 데이터 타입을 의미하는 수학적 함수로, 얼만큼 빠르게 one-dimensional 값이 애니메이션이 진행하는 동안 변경되는지를 기술한다.



온라인 데모

→ https://easings.net/ko

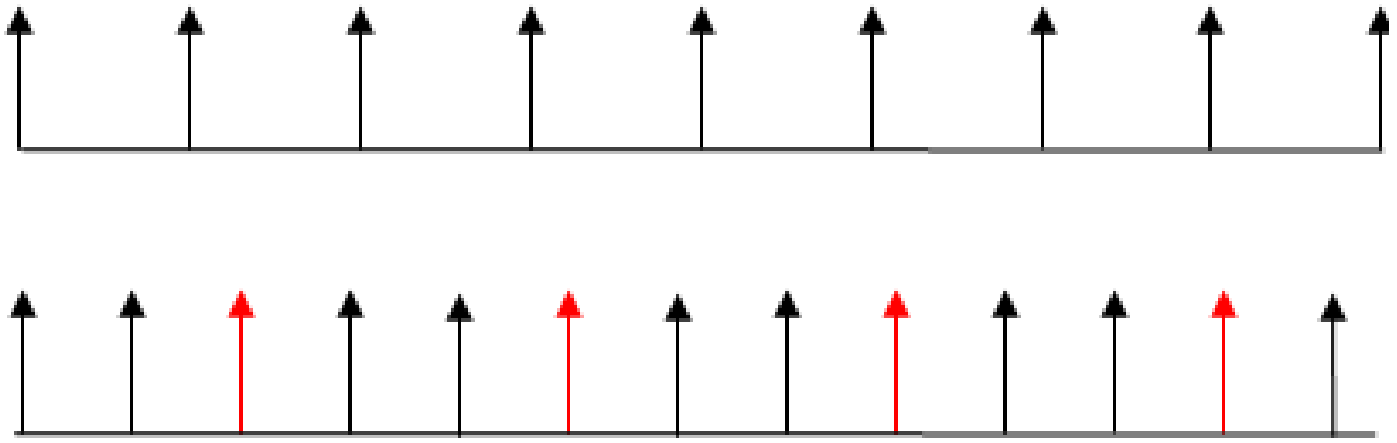
→ http://cubic-bezier.com/



requestAnimationFrame (rAF)

- 화면에 표시할 수 있는 최대 프레임 수 60FPS.(60/1000)
- 더 빨리 표시하면 건너뛰기 때문에 애니메이션이 끊김 현상이 발생하고 CPU가 낭비.

아래와 같이 10ms마다 timer을 호출하면 빨간색 부분이 건너뛰어 애니메이션이 끊김.



requestAnimationFrame

```
const ele = document.querySelector("#some");
let id;

function move() {
	const left = parseInt(ele.style.left) || 0;

	ele.style.left = (left + 5) + "px";

	if(left == 100) {
		cancelAnimationFrame(id);
	} else {
		id = requestAnimationFrame(move);
	}
}

id = requestAnimationFrame(move);
```

실습

- 다양한 방법의 애니메이션을 적용해 보자.

7. History API

History API

history 객체는 사용자 history에서의 앞 뒤 이동이 가능하도록 유용한 메서드와 속성들을 제공하며, history stack의 내용을 조작할 수 있게 한다.

SPA(Single Page Application)을 만들 때 주로 사용될 수 있다.

Methods

- back(): 이전 history로 이동
- forward(): 다음 history로 이동
- go(*number*): history 내의 특정 지점으로 이동
- pushState(): 새로운 history 상태를 history stack에 추가
- replaceState(): 현재 history 상태를 대체

Events

- hashChange 이벤트 : 주소창에 hash값이 변경될 때 발생하는 이벤트
- history 이벤트 : URL이 변경될 때 발생하는 이벤트

pushState()

history 객체에 새로운 history 상태를 추가한다.

```
const stateObj = {foo : "bar"};
history.pushState(stateObj, "page 2", "bar.html");
```

- state 객체: 직렬화(serializing) 할수 있는 객체
- title: history 타이틀
- URL: 추가되는 history 상태 주소 값

replaceState()

pushState()와 유사하나, 추가하지 않고 현재의 history 상태를 대체한다.

```
history.replaceState(stateObj, "page 3", "bar2.html");
```

현재 페이지에서 위의 코드를 콘솔에서 실행 후, 다른 페이지로 이동했다가 '뒤로가기' 버튼을 눌러 확인해 본다.

hashChange 이벤트

주소창에 hash값이 변경될 때 발생하는 이벤트

- 트위터등에서 사용하는 방법.

```
window.addEventListener('hashchange', function(event) {
        const old_url = event.oldURL; // 변경 전 url
        const new_url = event.newURL; // 변경 후 url
},false);
```

hashChange 이벤트

1) 브라우저에서 http://m.naver.com/ 이동
2) 개발자 콘솔에서 location.hash="#page=2" 실행
3) 개발자 콘솔에서 hashchange 이벤트 바인딩

```
window.addEventListener("hashchange", function(e) {
    console.log(e);
}, false);
```

4) 브라우저에서 뒤로가기 버튼을 눌러 'hashchange' 이벤트 발생 확인

→ **단점:** 파라미터를 파싱하고 관리하기가 귀찮을 수 있다.

popstate 이벤트

히스토리 값을 변경하는 이벤트로 hashChange 이벤트와 유사하다.

```
history.pushState(data, title, url);
// → 주소를 변경하는 메서드(데이터, 타이틀, url) - history stack에 쌓임

history.replaceState(data, title, url);
//→ 주소만 변경하는 메서드(데이터, 타이틀, url) - history가 바뀜

window.addEventListener("popstate", function(event) {
    // 주소가 변경되면 발생하는 이벤트
    // state에는 pushState, replaceState의 첫 번째 인자가 들어가 있다.
      event.state;
},false);
```

popstate 이벤트

1) 브라우저에서 http://m.naver.com/ 이동
2) 개발자 콘솔에서 history.pushState({"page": 2}, "title", "/page2") 실행
3) 개발자 콘솔에서 popstate 이벤트 바인딩

```
window.addEventListener("popstate", function(e) {
    console.log(e);
}, false);
```

4) 브라우저에서 뒤로가기 버튼을 눌러 'hashchange' 이벤트 발생 확인

popstate 이벤트

- URL기반으로 주소가 깔끔하게 처리할 수 있다.
- URL 디자인하기가 좋다.
- fallback을 구현하기 좋다.

8. Device Orientation Event

Device Orientation Event

- 기기의 움직임을 확인하는 이벤트로 deviceorientaion, devicemotion이 있음.
- 주로 간단한 게임이나 인터렉션에 많이 사용.

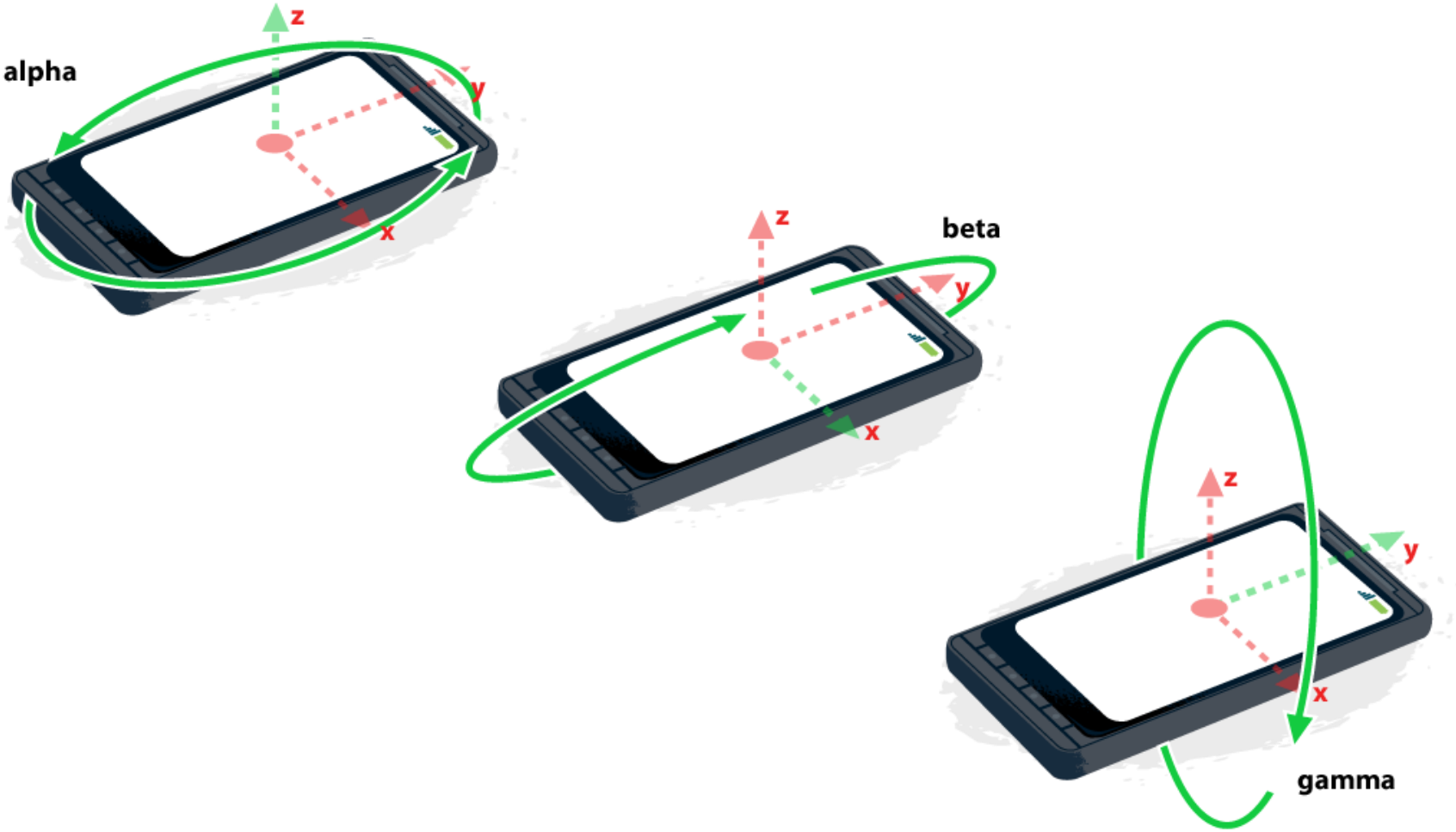
→ Spec : http://www.w3.org/TR/orientation-event/

Device Orientation Event

- **deviceorientation**
  실제 방향이 변경되면(예: 사용자가 장치를 기울이거나 회전한 경우) deviceorientation 이벤트가 창에서 실행되고 알파, 베타 및 감마 회전 각도(도 단위)를 제공

- **devicemotion**
  장치가 이동하거나 회전(보다 정확하게는 가속)한 경우에는 devicemotion 이벤트가 창에서 실행되고 X, Y, Z축의 가속도 (장치의 중력 가속도가 있는 경우와 없는 경우 모두 포함, 단위: m/s2) 및 알파, 베타, 감마 회전 각도 (단위: 도/초)의 변화량을 제공

Device Orientation Event



Device Orientation Event

```
window.addEventListener("deviceorientation", function (evt) {
        const directions = document.getElementById("directions");
        if (evt.alpha < 5 || evt.alpha > 355) {
                directions.innerHTML = "North!";
        } else if (evt.alpha < 180) {
                directions.innerHTML = "Turn Left";
        } else {
                directions.innerHTML = "Turn Right";
        }
}

window.addEventListener("devicemotion", function (evt) {
        const status = document.getElementById("status");
        const accl = evt.acceleration;
        if (accl.x > 1.5 || accl.y > 1.5 || accl.z > 1.5) {
                status.innerHTML = "EARTHQUAKE!!!";
        } else {
                status.innerHTML = "All systems go!";
        }
}
```

9. Screen Object

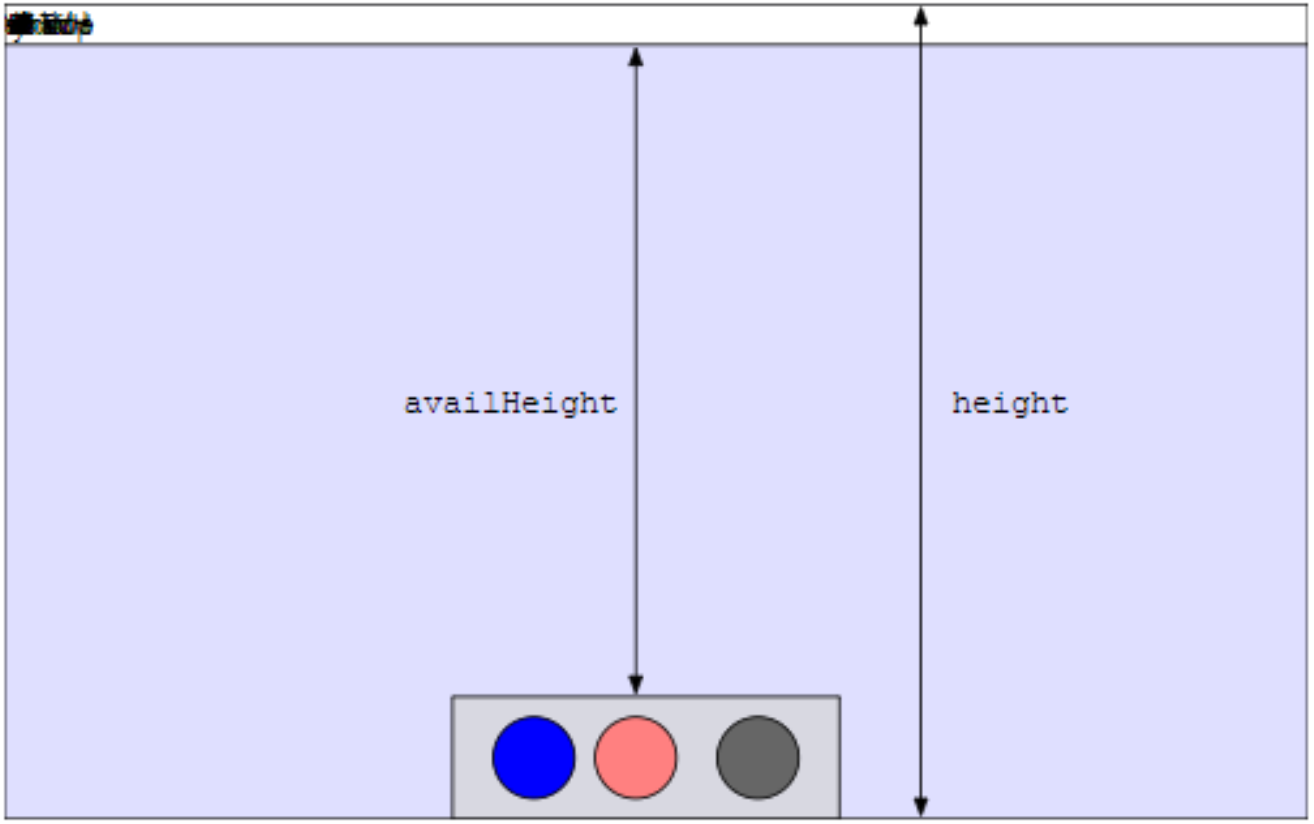
Screen Object

Scree 객체는 '스크린'에 대한 정보를 갖고 있는 객체로, 현재 스크린 상태에 대한 값을 얻을 수 있다.

속성	타입	설명
availTop	Number	사용자 인터페이스로 사용할 수 있는 스크린의 y 좌표 시작 값
availLeft	Number	사용자 인터페이스로 사용할 수 있는 스크린 좌측을 기점으로의 시작 값
availWidth	Number	Window에서 사용 가능한 가로축 픽셀 값
availHeight	Number	Window에서 사용 가능한 세로축 픽셀 값
colorDepth	Number	스크린의 color depth
pixelDepth	Number	스크린의 bit depth
width	Number	스크린의 너비 픽셀 값
height	Number	스크린의 높이 픽셀 값
orientation	Object	스크린의 방향에 대한 정보 객체 (Screen orientation API)

Screen Object: availHeight

MacOS의 경우, 화면 하단에 dock이 위치하게 되면 이때 availHeight의 값은 dock이 차지하는 영역과 상단 메뉴의 높이 값을 제외한 값이다.



Screen Orientation API

스크린의 orientation에 대한 정보를 반환한다.

```
const screenData = screen.orientation;

// 현재 문서의 orientation angle 값
screenData.angle;

// 현재 문서의 orientation 타입
// "portrait-primary", "portrait-secondary", "landscape-primary" 또는
"landscape-secondary".
screenData.type;

// 스크린의 orientation이 변경되었을 때 발생
screenData.addEventListener("change", function(e) {
      // 스크린의 방향이 변경됨
});
```

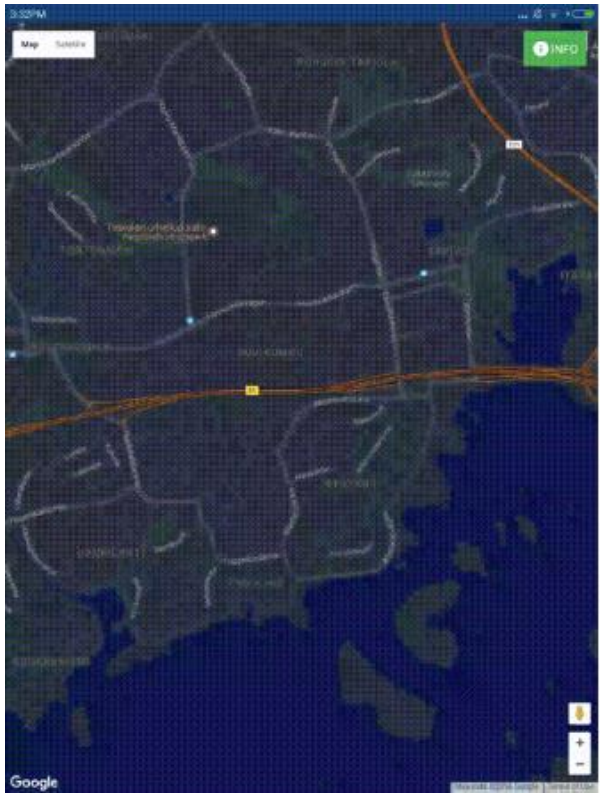
Screen Orientation: Type

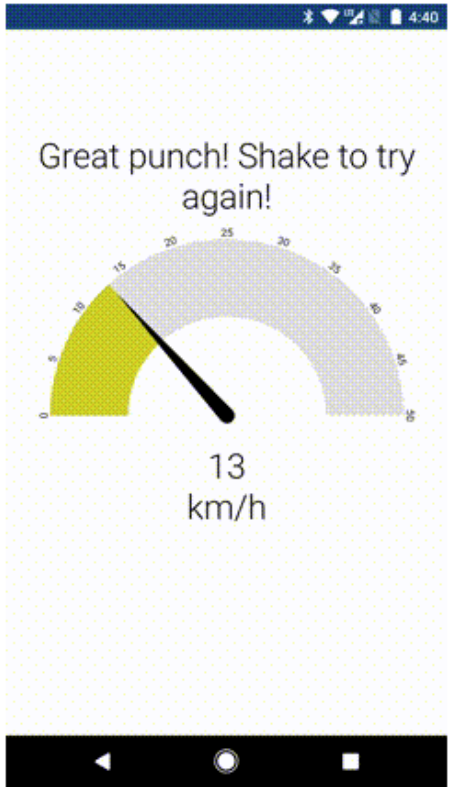
타입은 기기의 방향에 따라 다음과 같이 세부적인 값을 반환한다.

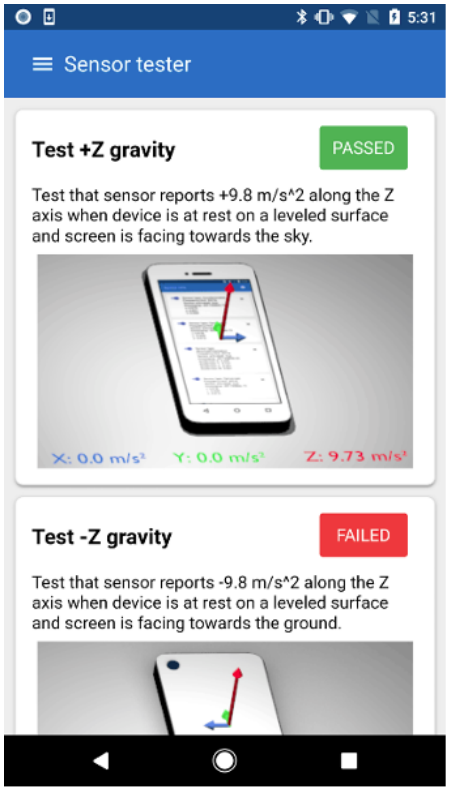
타입	설명
portrait-primary	스마트폰의 경우, 기기의 방향이 세로이면서 버튼이 아래에 위치한 경우
portrait-secondary	스마트폰의 경우, 기기의 방향이 세로이면서 버튼이 상단에 위치한 경우
landscape-primary	스마트폰의 경우, 기기의 방향이 가로이면서 버튼이 오른쪽에 위치한 경우
landscape-secondary	스마트폰의 경우, 기기의 방향이 가로이면서 버튼이 왼쪽에 위치한 경우

Generic Sensor API

휴대용 디바이스에서 사용 가능한 "sensor"들의 데모
→ https://intel.github.io/generic-sensor-demos/







고맙습니다.