# WebXR Device API 설명

WebXR Device API는 일반적으로 가상 현실 (VR)과, 증강 현실 (AR) 디바이스와 관련된 입력 및 출력 기능에 대한 액세스를 제공합니다. 웹에서 VR 및 AR을 개발하고 호스팅을 할 수 있습니다.

# WebXR이란?

## 목표

웹에서 XR 어플리케이션을 구동가능하게 다음을 수행하도록 허용합니다.

XR 기능을 지원하는지 검사

XR 기능을 쿼리

XR 장치 및 관련 입력 장치 상태를 등록

XR 장치에 적절한 프레임으로 이미지를 표시

## 부가 목표

AR / VR 웹이 작동하는 방식을 정의합니다.

AR / VR 하드웨어의 모든 기능을 가져옵니다.

[가상공간](https://en.wikipedia.org/wiki/Metaverse)을 구축합니다.

## 사용가능한 장치

* [ARCore 호환 장치](https://developers.google.com/ar/discover/supported-devices)
* [Google Daydream](https://vr.google.com/daydream/)
* [HTC Vive](https://www.htcvive.com/)
* [매직 리프 원](https://www.magicleap.com/magic-leap-one)
* [Microsoft Hololens](https://www.microsoft.com/en-us/hololens/)
* [오큘 러스 리프트](https://www3.oculus.com/rift/)
* [삼성 Gear VR](http://www.samsung.com/global/galaxy/gear-vr/)
* [Windows Mixed Reality 헤드셋](https://developer.microsoft.com/en-us/windows/mixed-reality)

## XR에서 X의 의미

변수로서의 X이다. x에는 V / A가 보통 들어가며, E(Extended) 나 C(Cross)도 가능함. 모든 게 가능

## OpenXR(<https://www.khronos.org/openxr>)과의 관계

기본 기능은 동일하나, 서로 다른 표준기관에서 개발하고 있는 별개의 API이다.

많은 동일한 개념들이 두 API에 의해 다른 방식으로 표현되므로, OpenXR이 상용화되면 WebXR은 이를 사용해 기능을 구현하는 것이 합리적일 것으로 예상함.

# 사용 사례

초기 XR 하드웨어를 게이머에게 마케팅하면 자연스럽게 이 API가 게임 개발에 주로 사용된다고 가정 할 수 있습니다. 밀접한 관련이 있는 WebGL API의 역사를 고려할 때 확실히 우리가 기대하는 것이지만, 대규모 게임보다 훨씬 더 "long-tail"스타일의 콘텐츠를 보게 될 것입니다. 일반적으로 웹의 XR 콘텐츠는 주요 VR / AR 하드웨어 제공 업체가 기본적으로 배포하고 있는 앱 스토어 형식에 알맞지 않거나 콘텐츠 자체가 스토어의 가이드라인에 맞지 않는 부분이 있습니다. 몇 가지 고수준의 예시는 다음과 같습니다.

## 비디오, 개체/데이터 시각화, 예술적 경험

개체/데이터 시각화

사이트는 WebXR을 통해 손쉬운 3D 시각화를 제공 할 수 있으며, 렌더링을 점진적으로 개선합니다. 3D 모델(예 : SketchFab ), 건축 사전 시각화, 의료 이미징, 매핑 및 기본 데이터 시각화를 보는 것은 영향력이 있고 이해하기 쉬우며, VR및 AR로 실사용 하는듯한 느낌을 줄 수 있습니다. 이러한 사용 사례의 경우, 특히 웹 콘텐츠가 단순히 링크이거나 클릭하면 되는 경우에 네이티브 앱 설치를 반길 사용자는 거의 없습니다.

홈쇼핑 애플리케이션 (예 : Matterport )은 특히 효과적인 데모(체험) 역할을 합니다. 장치 기능에 따라 사이트는 간단한 photo carousel부터 대화형 3D모델이 있는 화면을 걸어가며 VR로 볼 수 있는 등의 모든 방식으로 확장 할 수 있으므로 사용자에게 집에 실제로 있는듯한 느낌을 줍니다. 사용자에게 특별한 설치 없이 기능을 체험할 수 있게 하는 것은 사용자와 개발자 모두에게 좋습니다. 사용자가 무거운 (악성 일 수 있는) 실행 파일을 미리 설치하게 할 필요가 없기 때문입니다.

# VR 웹 앱의 구동방식

대부분의 WebXR 애플리케이션이 거치는 기본 단계는 다음과 같습니다.

1. 원하는 XR 모드가 지원되는지 **쿼리** 하십시오.
2. 지원이 가능한 경우 사용자에게 **XR 기능** 을 **알립니다**.
3. [사용자 활성화 이벤트는](https://html.spec.whatwg.org/multipage/interaction.html#activation) 사용자가 XR을 사용하고자 함을 나타냅니다.
4. 기기에서 **immersive 세션 요청**
5. 세션을 사용하여 XR 장치에 표시 할 그래픽 프레임을 생성 **하는 렌더링루프** 를 **실행합니다**.
6. 사용자가 XR 모드를 종료하겠다고 표시 할 때까지 계속 **프레임을 생성** 합니다.
7. **XR 세션을 종료합니다**.

다음 섹션에서 코드 예제는 먼저 immersive VR 세션을 사용하는 이 수명주기 시퀀스를 통해 핵심 API 개념을 보여준 다음 나중에 [인라인 세션](https://github.com/immersive-web/webxr/blob/master/explainer.md#inline-sessions) 에서 도입 된 차이점을 다룹니다. 코드 예제는 모두 동일한 응용 프로그램에 속하는 것으로 읽어야 합니다.

## XR 하드웨어

UA는 사용자에게 이미지를 제공 할 수 있는 XR 하드웨어의 사용 가능한 물리적 단위 (여기서는 "XR 장치"라고 함)를 식별합니다. 데스크톱 클라이언트에서는 일반적으로 헤드셋 주변 장치입니다. 모바일 클라이언트에서는 뷰어 장치 (예 : Google Cardboard / Daydream 또는 Samsung Gear VR)와 함께 모바일 장치 자체를 나타낼 수 있습니다. 또한 스테레오 표현 기능이 없지만 ARCore / ARKit 호환 장치와 같은 고급 추적 기능이 있는 장치를 나타낼 수도 있습니다. XR 기능 또는 기능에 대한 모든 쿼리는 이 장치에 대해 암묵적으로 수행됩니다.

비 표준 참고사항: 사용 가능한 XR장치가 여러 개인 경우 UA는 한 장치만을 선택해야 합니다. UA는 사용자가 장치 우선순위를 관리할 수 있는 설정 UI를 제공해, 장치 선택에 관한 모든 기준을 사용할 수 있습니다. Navigator.xr.isSessionSupported 혹은 navigator.xr.requestSession 을'inline' 에서 호출하는 것은 장치 선택UI를 건드리지 않습니다. 그러나 이는 많은 사이트들로 하여금 XR관련 대화상자를 사용자의 활성화 없이 문서 생명주기 초반에 나타나게 합니다.

처음에 사용할 수있는 XR 장치가 없더라도 응용 프로그램이 실행되는 동안 하나를 사용할 수 있게 되거나 이전에 사용 가능한 장치를 사용할 수 없게 될 수 있습니다. 이는 언제든지 연결하거나 분리 할 수 있는 PC 주변 장치에서 일반적인 상황이죠. 페이지는 페이지로드 후 장치 가용성의 변경에 응답하기 위해 navigator.xr에서 나온 devicechange 이벤트를 수신합니다. (페이지가 로드 될 때 이미 사용할 수 있는 XR 장치는 devicechange이벤트를 발생시키지 않습니다.) devicechange는 Event유형의 이벤트를 발생시킵니다.

navigator.xr.addEventListener('devicechange', checkForXRSupport);

## XR 기능 감지 및 광고

XR 장치와의 상호 작용은 XR Session 인터페이스를 통해 수행 되지만, XR 사용 페이지가 세션을 요청하기 전에 원하는 XR 콘텐츠 유형이 현재 하드웨어 및 UA에서 지원되는지 물어봐야 합니다. 그렇다면 페이지는 XR 기능을 사용자에게 광고 할 수 있습니다. (예를 들어, 사용자가 XR 콘텐츠를 시작하기 위해 시작버튼을 페이지에 추가합니다.)

이 navigator.xr.isSessionSupported 기능은 장치가 필요한 XR 기능을 지원하는지 확인하는 데 사용됩니다. XR Mode로 원하는 기능을 설명하고, 프로미스 결과를 반환하는데 성공적으로 XR Mode를 사용해 XR Session을 생성하면 Promise가 해결된다(return true). 그렇지 않으면 False를 반환한다. (프로미스는 try-catch와 비슷함.)

이러한 방식으로 요청하는 이유는 애플리케이션이 XR Session을 요청하기 전에 사용 가능한 XR Mode를 찾을 수 있도록 하기 위함이며, 이는 XR 장치 센서를 연결하고 프레젠테이션을 시작할 수 있게 해줍니다. 이로 인해 일부 시스템에서 상당한 전력 또는 성능 오버 헤드가 발생할 수 있으며 사용자의 화면을 가리거나 상태정보나 첫 화면을 따위를 실행시키고, XR 하드웨어에 액세스 된 다른 앱을 종료하는 등의 부작용이 있을 수 있습니다.

navigator.xr.isSessionSupported를 사용할 땐 시스템에서 실행중인 XR 응용 프로그램을 방해하거나 사용자에게 보이는 부작용이 없어야 합니다.

XR Mode는 두 가지가 있습니다.

## Inline : ‘inline’ 열거형 모드로 요청됩니다. Inline sessions는 XR장치에 콘텐츠를 표기하는 기능이 없지만 장치 추적정보에 액세스하고 이를 사용하여 페이지에서 콘텐츠를 렌더링하는 데 사용할 수 있습니다. (페이지에 렌더링 된 Scene이 장치의 이동에 반응하는 이 기술을 "Magic Window" 모드라고도 합니다.) WebXR 장치 API를 구현하는 UA는 페이지 기능 정책에 의해 차단되지 않는 한 XR 장치 존재 여부에 관계없이 Inline sessions이 생성 될 수 있음을 보장해야 합니다.

## Immersive VR : ‘immersive-vr’열거형 모드로 요청됩니다. Immersive VR 콘텐츠는 XR 장치에 직접 표시됩니다. (예: VR헤드셋에 표시됨) Immersive VR Session은 Immersive session의 요청을 허용하도록 하는 또 다른 콜백이나 사용자 활성화 이벤트 내에서 요청해야 합니다.

Immersive = 몰입 형

다음 예에서는 몰입 형 VR 세션을 사용하는 핵심 API 개념을 먼저 설명하고 나중에 인라인 세션에서 도입 된 차이점을 다룹니다. 이를 염두에 두고 이 코드는 헤드셋과 같은 장치에 이미지를 표시하는 기능을 원하기 때문에 몰입 형 VR 세션의 지원을 확인합니다.

async function checkForXRSupport() {

// 몰입 형 VR 프레젠테이션을 지원하는 사용 가능한 XR 장치가 있는지 확인합니다.

// (예: 헤드셋에 표시), 기기에 기능이 있는 경우 페이지는 몰입 형 VR 콘텐츠를

// 시작하는 페이지에 “VR Enter” 버튼을 추가하려고 합니다.

navigator.xr.isSessionSupported('immersive-vr').then((supported) => {

if (supported) {

var enterXrBtn = document.createElement("button");

enterXrBtn.innerHTML = "Enter VR";

enterXrBtn.addEventListener("click", beginXRSession);

document.body.appendChild(enterXrBtn);

} else {

console.log("Session not supported: " + reason);

}

});

}

## 세션 요청

navigator.xr.isSessionSupported()로 원하는 모드를 사용할 수 있는지 확인 했으면,

어플리케이션은 XR 장치의 프레젠테이션 또는 추적 기능과 상호 작용하기 위해

navigator.xr.requestSession()메서드로 XR Session 인스턴스를 요청해야 합니다.

function beginXRSession() {

// 몰입 형 세션을 요청할 때 클릭 또는 터치와 같은 사용자 동작 이벤트에서 호출돼야 합니다.

navigator.xr.requestSession('immersive-vr')

.then(onSessionStarted)

.catch(err => {

// 여러 이유로 실패할 수 있습니다. 아마 이 시점에서

// 아무런 추적도 없이 그냥 정상적으로 그 장면을 렌더링하고 싶을 것입니다.

window.requestAnimationFrame(onDrawFrame);

});

}

이 샘플에서는 이전 샘플의 'VR Enter' 버튼을 클릭하여 실행한다고 가정하는 'begin XR Session' 기능이 'immersive-vr' 모드로 작동하는 'XRSession'을 요청한다. requestSession 메소드는 성공 후 "XRSession"으로 넘어가는 Promise을 반환한다. 개발자는 "XRSessionMode" 외에도 반환된 세션이 가져야 하는 기능이 포함된 "XR Session Init" 사전을 제공할 수 있다. 자세한 내용은 [기능 종속성](https://github.com/immersive-web/webxr/blob/master/explainer.md#feature-dependencies)을 참조하십시오.

"isSessionSupported"로 지정된 모드에 대한 Promise가 true로 해결된 경우, 외부 요인(예: 몰입 형 세션의 사용자 활성화 이벤트에서 session이 호출되지 않는 경우)을 제외하고 동일한 모드의 세션 요청이 성공할 것으로 합리적으로 예상해야 한다. UA는 궁극적으로 이 요청을 이행할 수 있는지 여부를 판단할 책임이 있다.

전체 UA에서 XR 장치당 하나의 몰입 형 세션만 허용됩니다. 몰입 형 세션이 요청되고 UA에 이미 활성화된 몰입 형 세션 또는 보류중인 몰입 형 세션의 요청이 있는 경우 새 요청을 거부해야 합니다. 몰입 형 세션이 활성화되면 모든 인라인 세션이 일시 중지 됩니다. 인라인 세션은 명시적으로 필요로 하는 다른 옵션과 쌍을 이루지 않는 한 사용자 활성화 이벤트 내에서 생성 할 필요가 없습니다.

세션이 시작되면 렌더링을 준비하기 위해 몇 가지 설정을 수행해야합니다.

"XRReferenceSpace"를 생성하여 "XRViewerPose"데이터가 정의될 공간을 확보해야 한다. 자세한 내용은  [Spatial Tracking Explainer](https://github.com/immersive-web/webxr/blob/master/spatial-tracking-explainer.md)를 참조하십시오.

'XRWebGLLayer'를 생성하여 'XRSession'의 'renderState.baseLayer'로 설정해야 함

('baseLayer'는 규격의 향후 버전이 다중 레이어를 활성화할 가능성이 높기 때문)

그런 다음 'XRSession.requestAnimationFrame'을 호출하여 렌더 루프 펌핑을 시작해야 한다.

let xrSession = null;

let xrReferenceSpace = null;

function onSessionStarted(session) {

// Store the session for use later.

xrSession = session;

xrSession.requestReferenceSpace('local')

.then((referenceSpace) => {

xrReferenceSpace = referenceSpace;

})

.then(setupWebGLLayer) // Create a compatible XRWebGLLayer

.then(() => {

// Start the render loop

xrSession.requestAnimationFrame(onDrawFrame);

});

}

https://github.com/immersive-web/webxr/blob/master/explainer.md#setting-up-an-xrwebgllayer