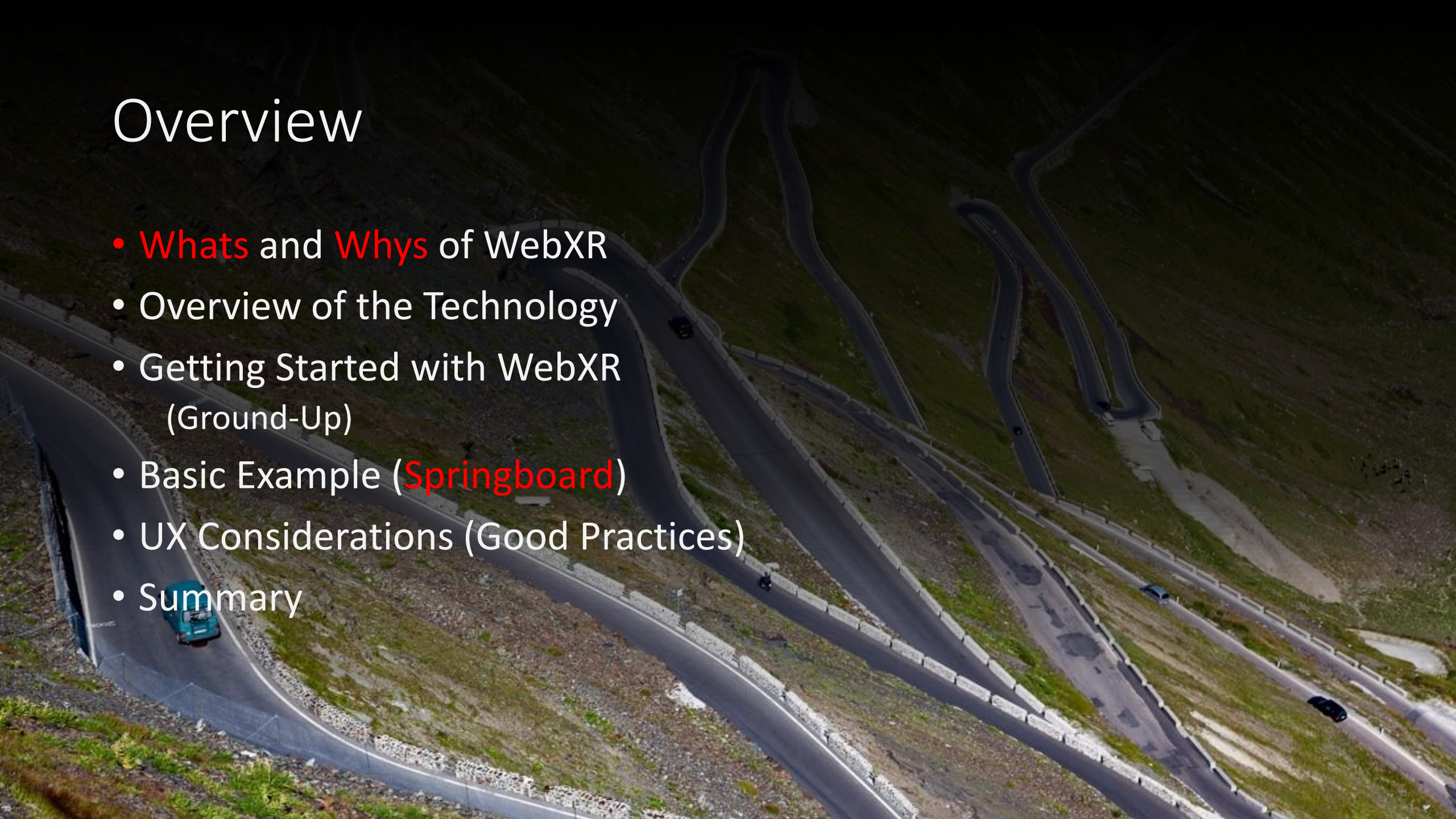
SIGGRAPH 2021

INTRODUCTION TO WEBXR

KENWRIGHT

Overview

- Whats and Whys of WebXR
- Overview of the Technology
- Getting Started with WebXR
  (Ground-Up)
- Basic Example (Springboard)
- UX Considerations (Good Practices)
- Summary



What is WebXR?

What is WebXR?

- Cross Reality?
- Extreme Reality?
- Extensible Reality?
- ‘Whatever You Want’ Reality!

‘X’

WebXR is a group of standards which are used together to support hardware designed for all types of XR technologies.

The ‘XR’ in WebXR stands for extended reality. The ‘X’ is meant as a variable – augmented reality (AR), virtual reality (VR) and mixed reality (MR) are all types of XR.

What is WebXR?

- WebXR Device API Specification

- Frictionless access
- W3C standard
- Widely supported
- Works everywhere

  (Cross-Platform)

- eXtended Reality
  - Augmented/Mixed/Virtual Reality

  (More than just ‘Visual’)
  - Sensors
  - Gamepads

Why did we need WebXR?

Why did we need WebXR?

- The landscape is evolving fast!
- Consistent standards
- Lots of Advances in XR Technologies





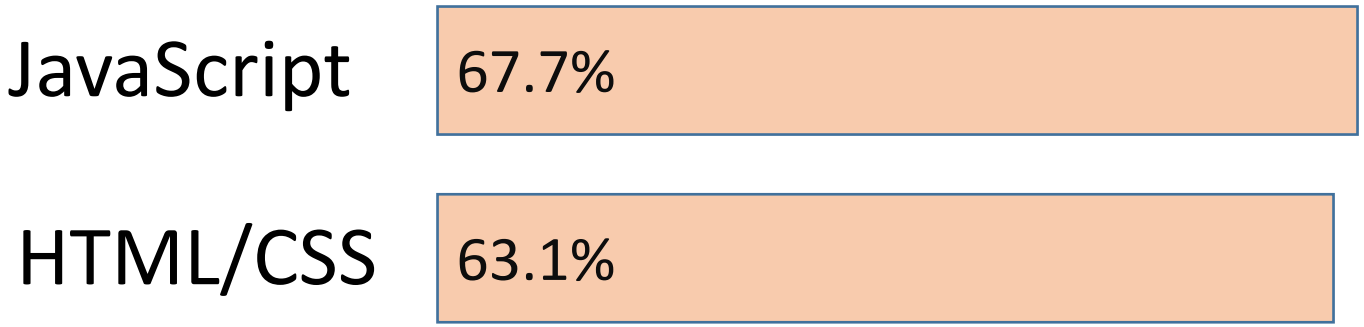


Web Technologies (Ideal)

Popular to Many

(2020 Developer Survey)

Unsurprisingly, for the eighth year in a row, JavaScript has maintained it's stronghold as the most commonly used programming language.



Top 2 most popular technologies (Stack Overflow Survey 2020)

WebXR Let’s You Dive Right In!

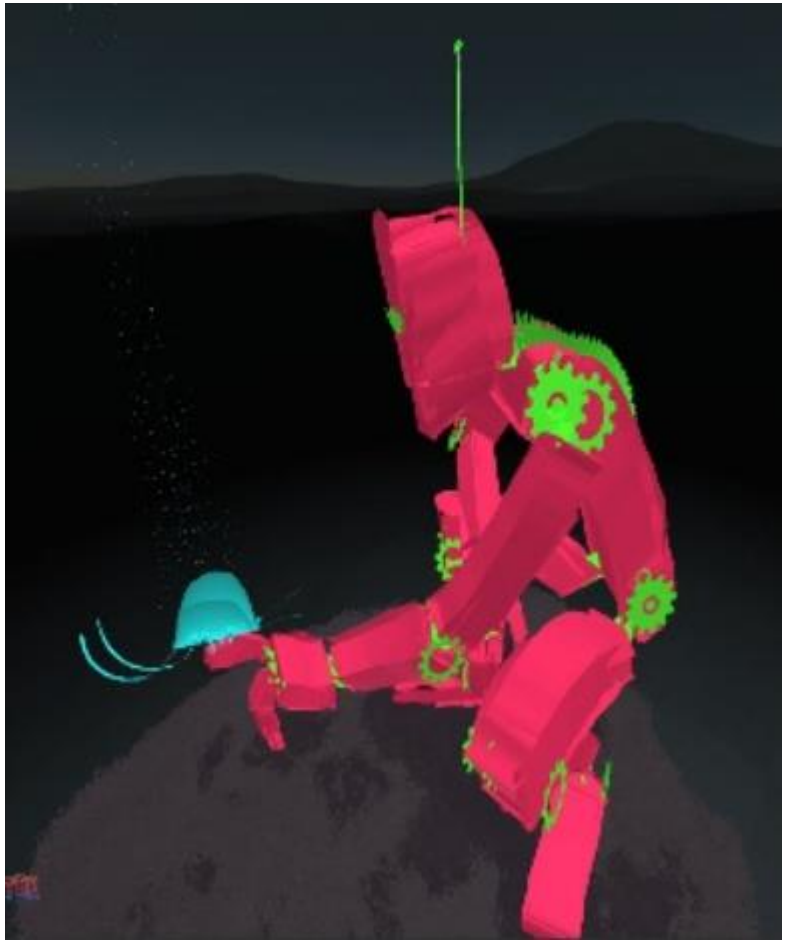
Runs in Browser

Connect with other Browser API/Resources (GPS/Location/Compass)

Is WebXR Just for Entertainment/Games?

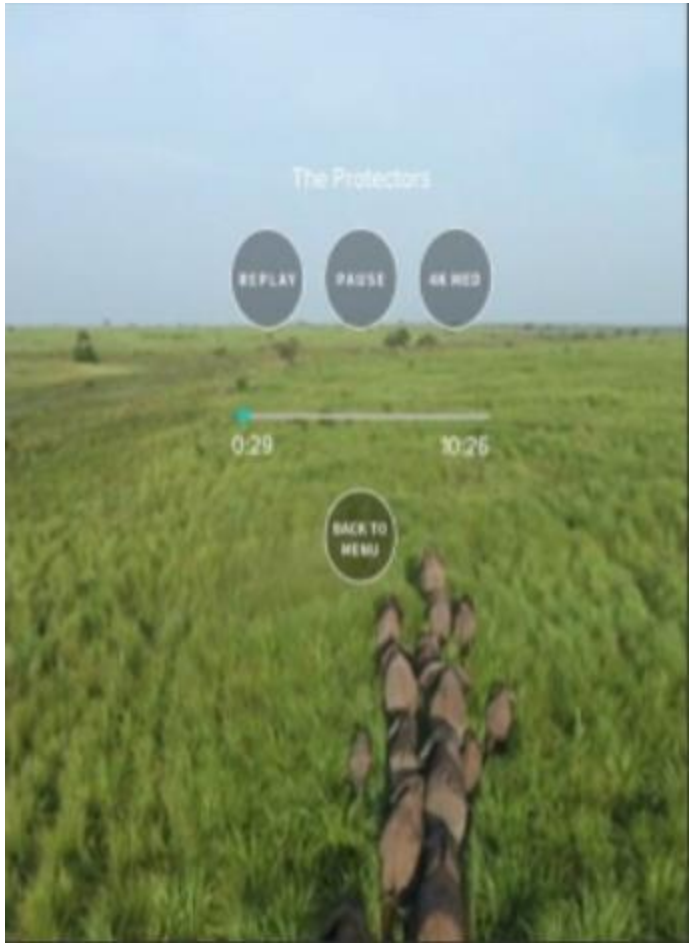
Is WebXR Just for Entertainment/Games?

ART



EDUCATION

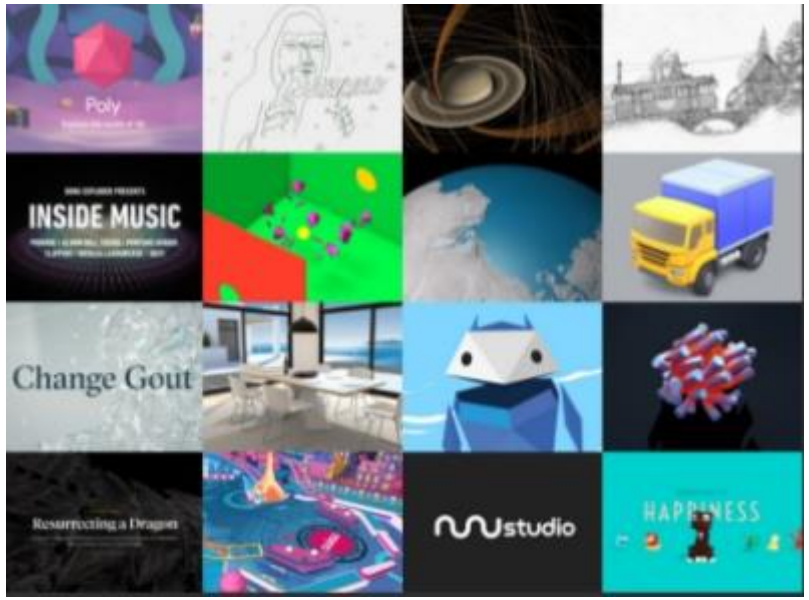
STORY-TELLING



Lots of Help/Support/Resources/Libraries

(Don’t have to start from nothing/ground-up)

For example three.js



aframe.io

Lots of others babylonjs.com, Unity, …

Embraced by Communities/Developers

Today Vanilla WebXR

Looking at the WebXR API

Working with WebXR Directly (No Hiding)

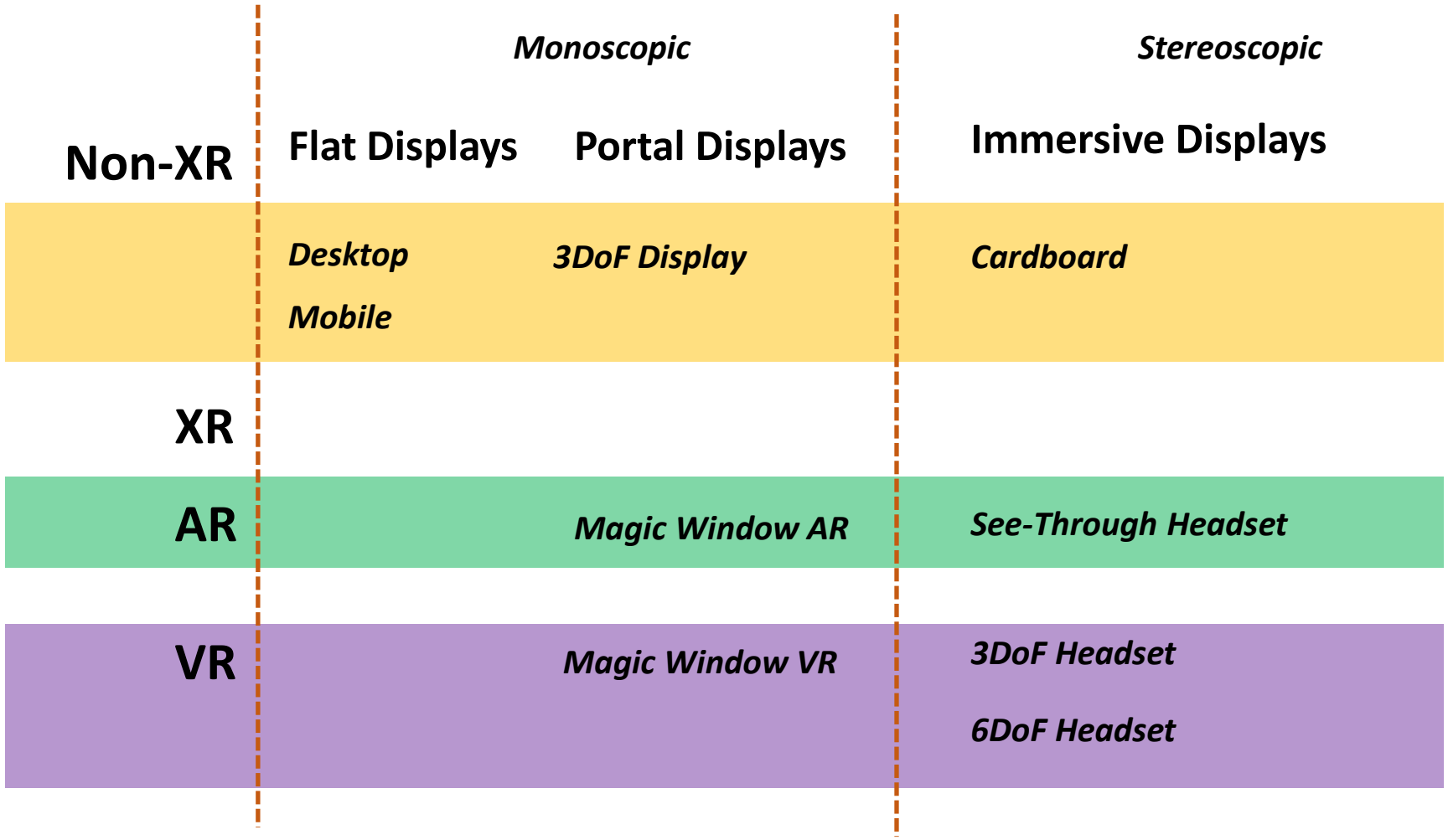
What's happening underneath

WebXR

Brings together the digital ‘virtual’ worlds (merging)

Mixed Reality (MR)

XR



WebXR Display Realities

Isn’t AR and MR Same?

Isn’t AR and MR Same?

NO

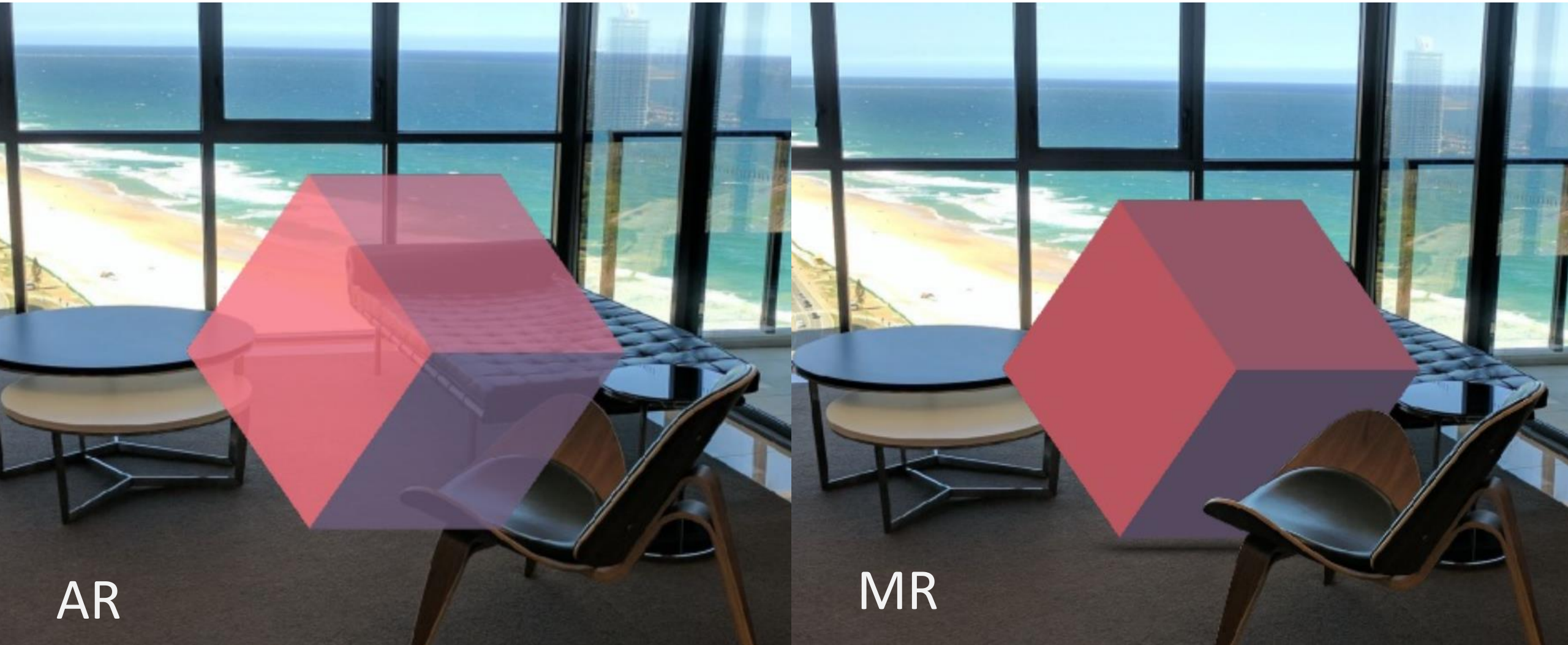
- Augmented Reality (AR)

'overlays' virtual objects on the real-world environment

- Mixed Reality (MR)

not just overlays but 'anchors' virtual objects to the real world

Isn’t AR and MR Same?



Support WebXR?

Increasing Support

https://caniuse.com/webxr

Small Details

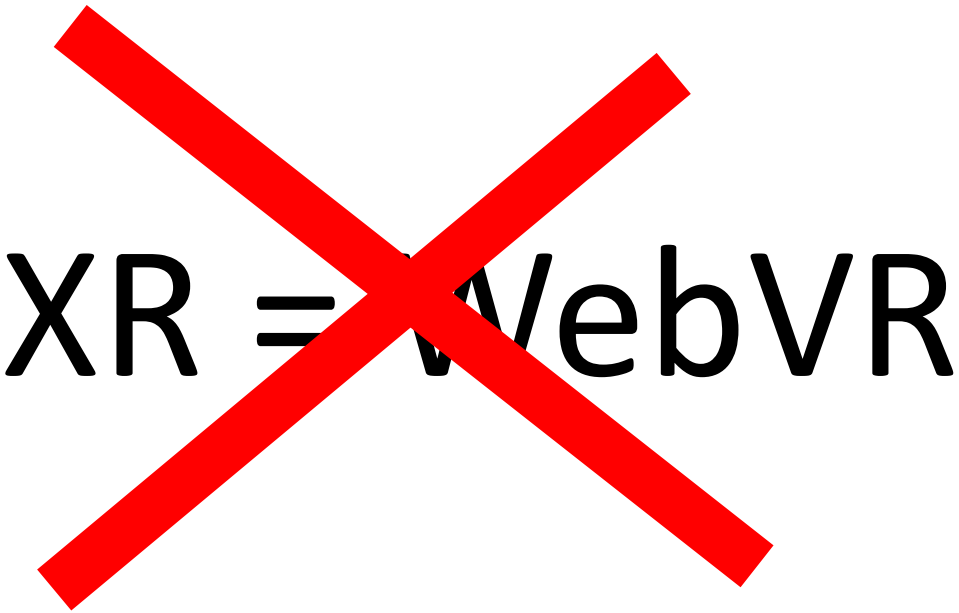
- Requires HTTPS
- Code in Browser
- Progressive VR/AR
  - “Write once, run anywhere”
  - Accessible across a much wider range of supported devices

WebXR Device API
(What does it do?)

- Detect AR/VR Devices
- Get Device Capabilities
- Get Device Orientation/Position
- Display Images/Graphics to the Frame Rate

WebVR API
(Is WebXR an Extension of WebVR?)

WebVR API
(Is WebXR an Extension of WebVR?)



WebVR to WebXR

- Not backward compatible with WebVR
- However, most WebVR apps ‘should port easily’
- Easy to upgrade/transition

Example

```
// WebVR
navigator.getVRDisplays(displays => { ...
// WebXR
navigator.xr.requestDevice().then(device => {
```

WebXR Device API

- Supports Augmented Reality
- Unified Input Model
- Clean Consistent, Predictable
- Better Browser Optimizations

WebXR Device API

https://www.w3.org/TR/webxr/

- **Device Enumeration**
  - XR
  - XRDevice
- **Session**
  - XRSessionCreationOptions
  - XRSession
  - Animation Frames
  - The XR Compositor
- **Frame Loop**
  - XRPresentationFrame
- **Coordinate Systems**
  - XRPresentationFrame
  - XRFrameOfReference
  - XRStageBounds
- **Views**
  - XRView
  - XRViewport
- **Pose**
  - Matrices
  - XRDevicePose
- **Input**
  - XRInputSource
  - XRInputPose
- **Layers**
  - XRLayer
  - XRWebGLLayer
  - WebGL Context Compatiblity

WebXR Steps

1. Request XR Device
2. Reveal XR Functionality
3. Request XR Session
4. Run Render Loop

https://codesandbox.io/s/basic-webxr-u4m2e

Request XR Device

```
/** 1 - called after page loads **/
function initXR() {
        if (!window.isSecureContext) {
          // has to be https!
          console.log("WebXR unavailable due to insecure context“);
        }

        if (navigator.xr) {
            // does this browser support/have webxr?
            xrButton.addEventListener("click", onButtonClicked);
            navigator.xr.addEventListener("devicechange", checkSupportedState);
            checkSupportedState();
        }
}
```

Check XR Session Support

```
/** 2 (just enables the button)**/
function checkSupportedState() {
        navigator.xr.isSessionSupported("immersive-ar").then((supported) => {
          if (supported) {
            xrButton.innerHTML = "Enter AR";
          } else {
            xrButton.innerHTML = "AR not found";
          }

          xrButton.disabled = !supported;
        });
}
```

Request XR Session

```
/** 3 **/
function onButtonClicked() {
        if (!xrSession) {
          const options = {
            // optionalFeatures: ['dom-overlay'],
            // domOverlay: {root: document.body}
          };
          navigator.xr
            .requestSession("immersive-ar")
            .then(onSessionStarted, onRequestSessionError);
        } else {
          xrSession.end();
        }
}
```

Start XR Session

```
/** 4 **/
function onSessionStarted(session) {
    xrSession = session;
    console.log("dom overlay is ", session.domOverlayState);
    xrButton.innerHTML = "Exit AR";

    // Show which type of DOM Overlay got enabled (if any)
    document.getElementById("session-info").innerHTML =
        "DOM Overlay type: " + session.domOverlayState.type;

    session.addEventListener("end", onSessionEnded);
    let canvas = document.createElement("canvas");
    gl = canvas.getContext("webgl", { xrCompatible: true });

    session.updateRenderState({ baseLayer: new XRWebGLLayer(session, gl) });
    session.requestReferenceSpace("local").then((refSpace) => {
      xrRefSpace = refSpace;
      session.requestAnimationFrame(onXRFrame);
    });
}
```

Setup an XRLayer

Start Render Loop

```
/** 5 **/
function onXRFrame(t, frame) {
      let session = frame.session;
      session.requestAnimationFrame(onXRFrame);

      let pose = frame.getViewerPose(xrRefSpace);
      if (pose) {
        gl.bindFramebuffer(gl.FRAMEBUFFER, session.renderState.baseLayer.framebuffer );

        const width = session.renderState.baseLayer.framebufferWidth;
        const height = session.renderState.baseLayer.framebufferHeight;
        gl.enable(gl.SCISSOR_TEST);
        gl.scissor(width / 4, height / 4, width / 2, height / 2);
        let time = Date.now();
        // void gl.clearColor(red, green, blue, alpha);
        gl.clearColor( Math.cos(time / 3000), Math.cos(time / 2000), Math.cos(time / 8000), 0.5 );
        gl.clear(gl.COLOR_BUFFER_BIT | gl.DEPTH_BUFFER_BIT);
      }
    }
```

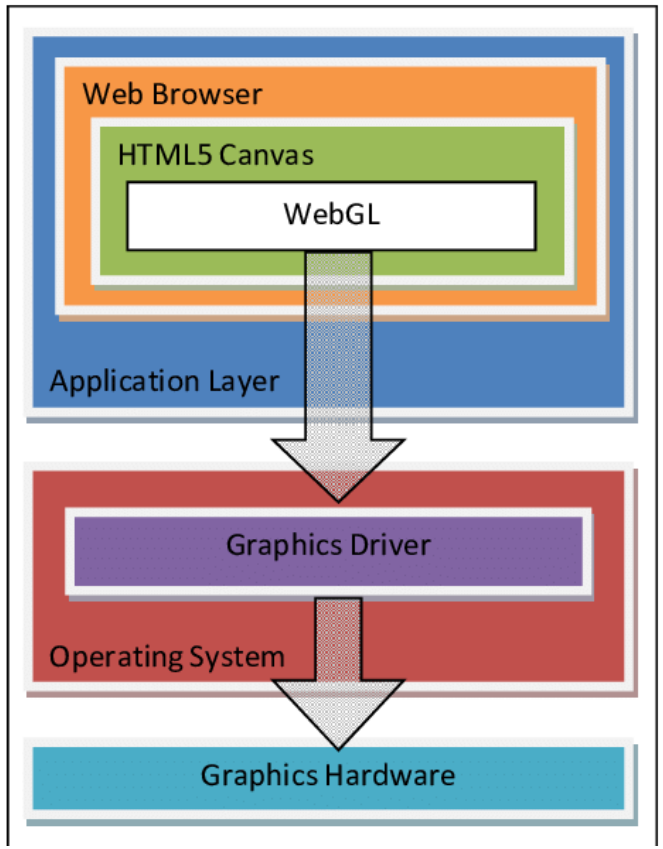
WebGL

Exit WebXR Session

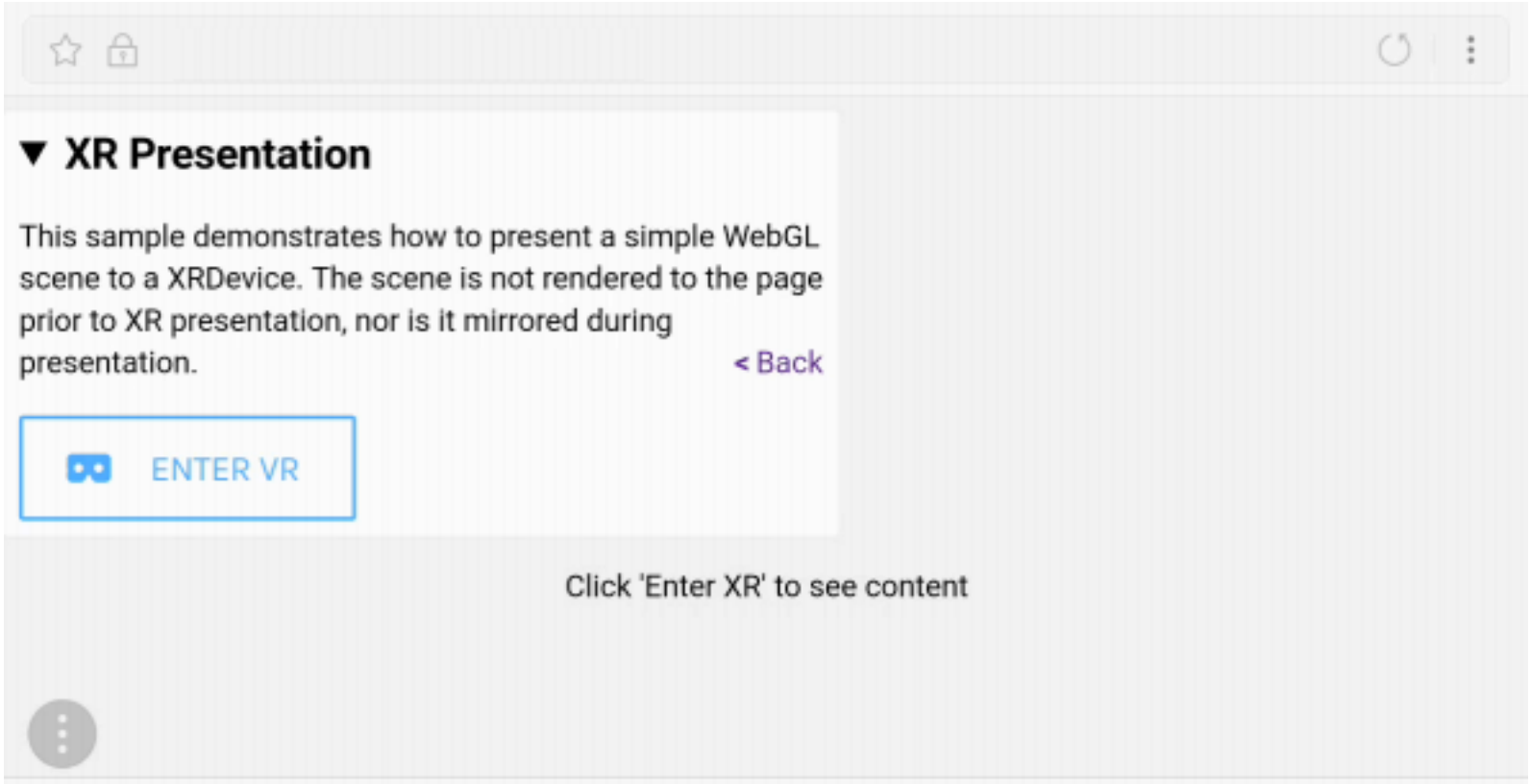
```
function onRequestSessionError(ex) {
    alert("Failed to start immersive AR session.");
    console.error(ex.message);
}

function onEndSession(session) {
    session.end();
}

function onSessionEnded(event) {
    xrSession = null;
    xrButton.innerHTML = "Enter AR";
    gl = null;
}
```

WebGL

- WebGL Graphics
- GPU/Hardware Accelerated
- Complete Control (Visuals)
- Transforms/User Input
  (Controllers/Headset)



What if no XR device is available?



Fallback (WebGL)

Render to the default output (Window/Canvas)

(Magic Window)

UX Design for WebXR Apps *(Good-Practises)*

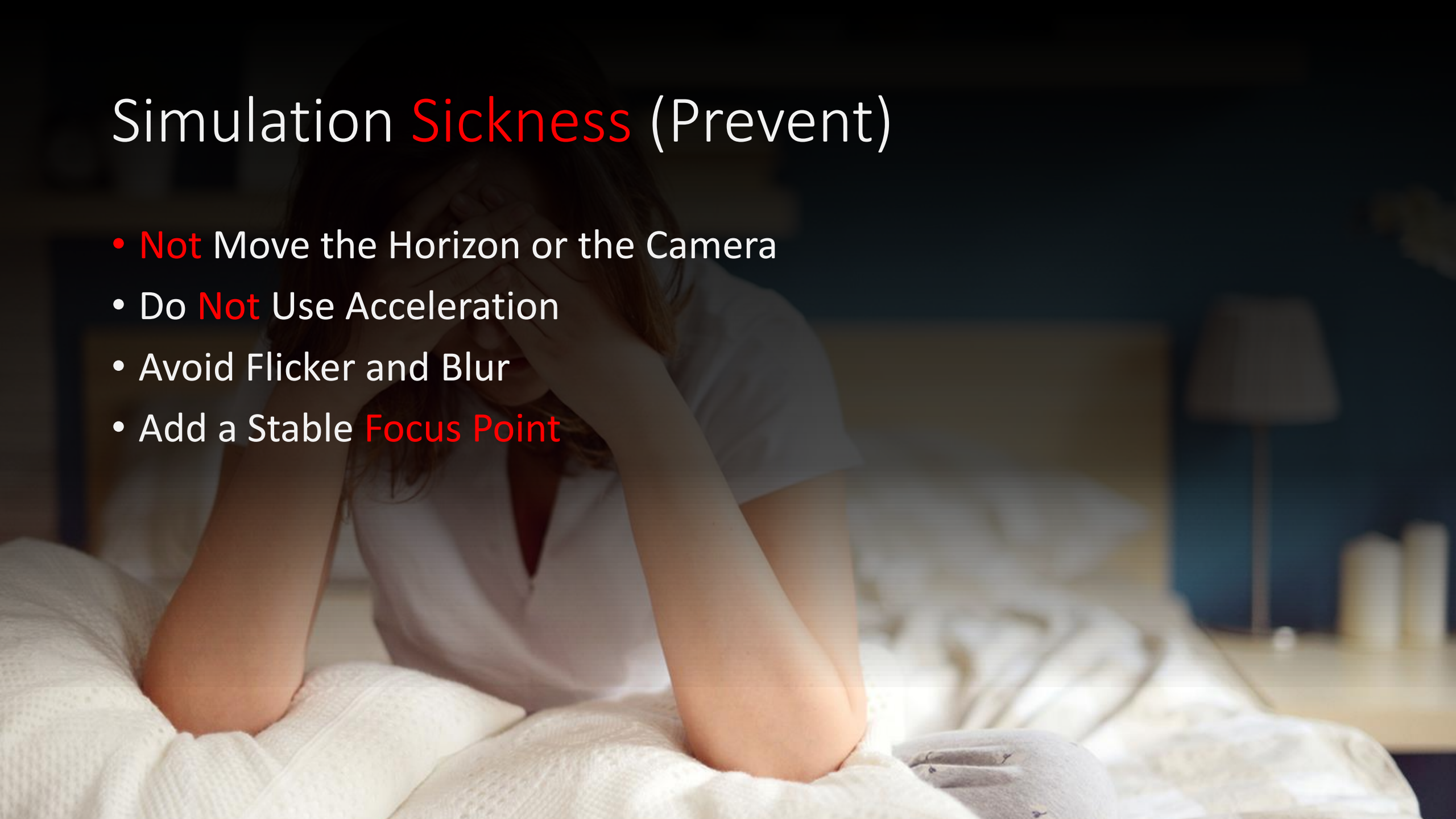
- 2D Interactive WebGL App vs XR App

- Add Feedback, Respond Immediately
- Guide Users with Cues
- Provide Information in Context

Comfort and Safety

- There is No Optimal Environment
- Consider a User's Movement
- Avoid Fatiguing your Users
- Do Not Trigger Phobias
- Do Not Move Things Fast Towards the Camera
- Respect a User's Safe Space

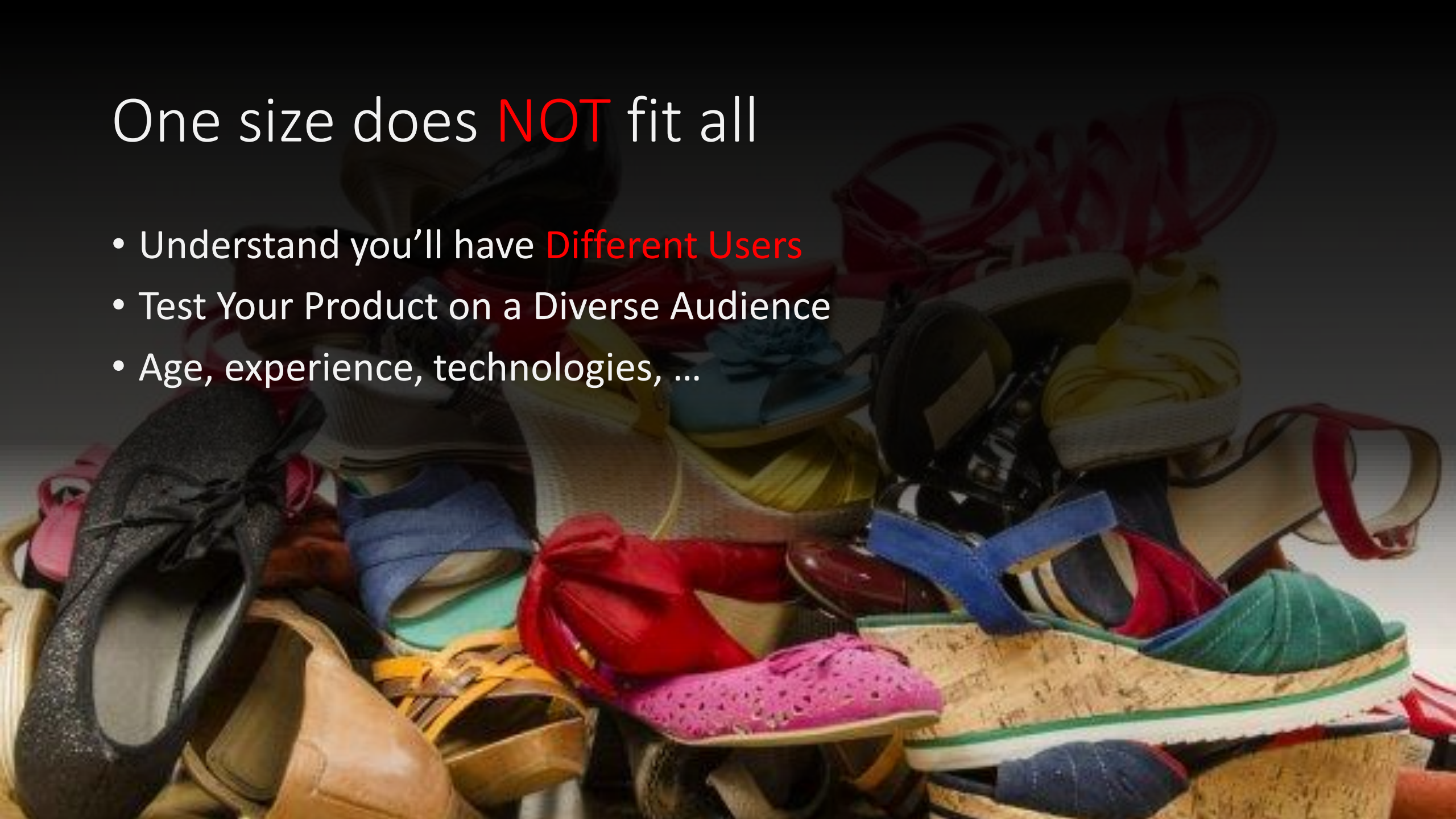
Simulation Sickness (Prevent)



- Not Move the Horizon or the Camera
- Do Not Use Acceleration
- Avoid Flicker and Blur
- Add a Stable Focus Point

One size does NOT fit all

- Understand you'll have Different Users
- Test Your Product on a Diverse Audience
- Age, experience, technologies, …



Apple – Human Interface Guidelines (AR)

https://developer.apple.com/design/human-interface-guidelines/ios/system-capabilities/augmented-reality/

Augmented Reality

Augmented reality (or AR) lets you deliver immersive, engaging experiences that seamlessly blend virtual objects with the real world. Using the device's camera to present the physical world onscreen live, your app superimposes three-dimensional virtual objects, creating the illusion that these objects actually exist. Depending on the experiences your app offers, people can reorient the device to explore the objects from different angles, interact with objects using gestures and movement, and even join other people in multiuser AR experiences. For developer guidance, see ARKit.

Offer AR features only on capable devices. If your app's primary purpose is AR, make your app available only to devices that support ARKit. If your app includes features that require specific AR capabilities, or if AR features are optional in your app, don't show people an error if they try to use these features on a device that doesn't support them; instead, simply avoid offering the feature on an unsupported device. For developer guidance, see Verifying Device Support and User Permission.

Creating an Engaging, Comfortable Experience


We’re at the beginning of a
new era of computing
WebXR is A New Dimension For Users and The Web

Get Involved

- Immersive Web at W3C
  - https://github.com/immersive-web/
- Immersive Web Working Group
  - https://www.w3.org/immersive-web/

Conclusion

- WebXR Here Now
- Easy to get started (few lines of JavaScript)
- Don’t need specialist hardware to give it try
- Lots of resources/libraries/tools/examples
- Goes beyond simple ‘Entertainment’ value
- Start experimenting and exploring!

  (fun)

Resources

- WebXR Coffee Break Series
- WebXR Device API spec
- WebXR Device API Explainer
- WebXR Samples
- WebXR Immersive WG
- WebXR Device API Github