Create Web AR Experiences

Fivos Doganis
fivos.doganis@gmail.com

Focus on standalone AR

- Native AR
  - ARCore
  - ARKit
- Web AR
  - WebXR ★
  - Alternatives
    - AR.js (marker, GPS)
    - MindAR (image, face)
    - and more!

cover picture: Artem Bryzgalov https://unsplash.com/@abrizgalov

Setup: testing on your mobile

- Check that your smartphone can read QR codes
- **iOS**
  - default Camera app
- **Android**
  - use Google Chrome + scan button
  - or install a **trustworthy** QR code scanning app like **Trend Micro**
- Other 100% web based alternatives
  - webqr.com
  - qrcodescan.in

Native AR

ARCore 

ARKit 

ModelViewer

Check if your device supports Native AR

- on your **mobile**, open

https://modelviewer.dev/

- click on the AR icon
- see the astronaut in AR 🎉
  - uses SceneViewer (Android)
  - or QuickLook (iOS)

Android

http://storage.googleapis.com/ar-answers-in-search-models/static/mandalorian/grogu/grogu.glb

- **GLB** file format
- Native **SceneViewer**
  - uses Google's **ARCore**
  - realistic lighting
- ARCore supported devices

iOS

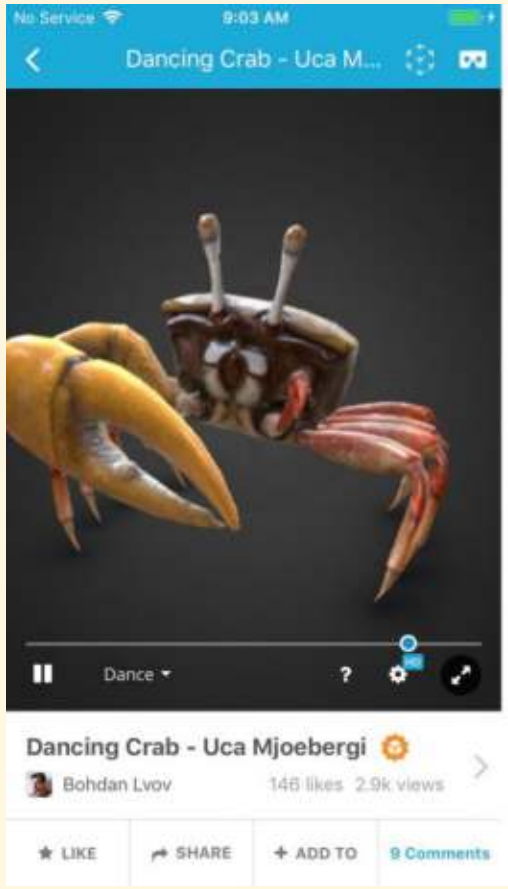
http://storage.googleapis.com/ar-answers-in-search-models/static/mandalorian/grogu/grogu.usdz

- **USDZ** file format
- Native AR **QuickLook Preview**
  - uses Apple's **ARKit**
  - realistic lighting and occlusions

More examples

https://sketchfab.com/

- to hide navigation bar, add https://www.sketchfab.com on your mobile's Home Page



WebAR

- WebXR
- Alternatives

Why use the Web for AR?

- **mobile** experiences
- **open** technologies
  - **cross-platform**
  - **non-propietary** (unlike Unity or Unreal)
  - **free**
  - **distribute by sharing URLs : no installation, no app store**
- easy integration with many existing Web APIs
  - **anchors the the web to the real world**
  - advanced **interactions**

WebXR

- Stack
- Setup
- Concepts
- WebXR AR API
- Code

Reminder: WebGL Stack

WebXR ↔ **WebGL** / WebGPU

OpenXR / Reality Kit / ARKit ↔ **OpenGL** / DirectX / Metal

HMD + controllers ↔ GPU + screen

WebGL Stack

Content downloaded from the Web

Content
JavaScript, HTML, CSS, ...

Middleware provides accessibility for non-expert programmers
E.g. three.js library

JavaScript Middleware
three.js babylon.JS PLAYCANVAS

Low-level WebGL API provides a powerful foundation for a rich JavaScript middleware ecosystem

Browser provides WebGL 3D engine alongside other HTML5 technologies - no plug-in required

WebGL | CSS | JavaScript | HTML5

OS Provided Drivers
WebGL uses native OpenGL or OpenGL ES or Angle = OpenGL ES over DX9/11

OpenGL|ES | OpenGL

Reliable WebGL relies on work by both GPU and Browser Vendors
->
Khronos has the right membership to enable that cooperation

WebGL architecture: software stack

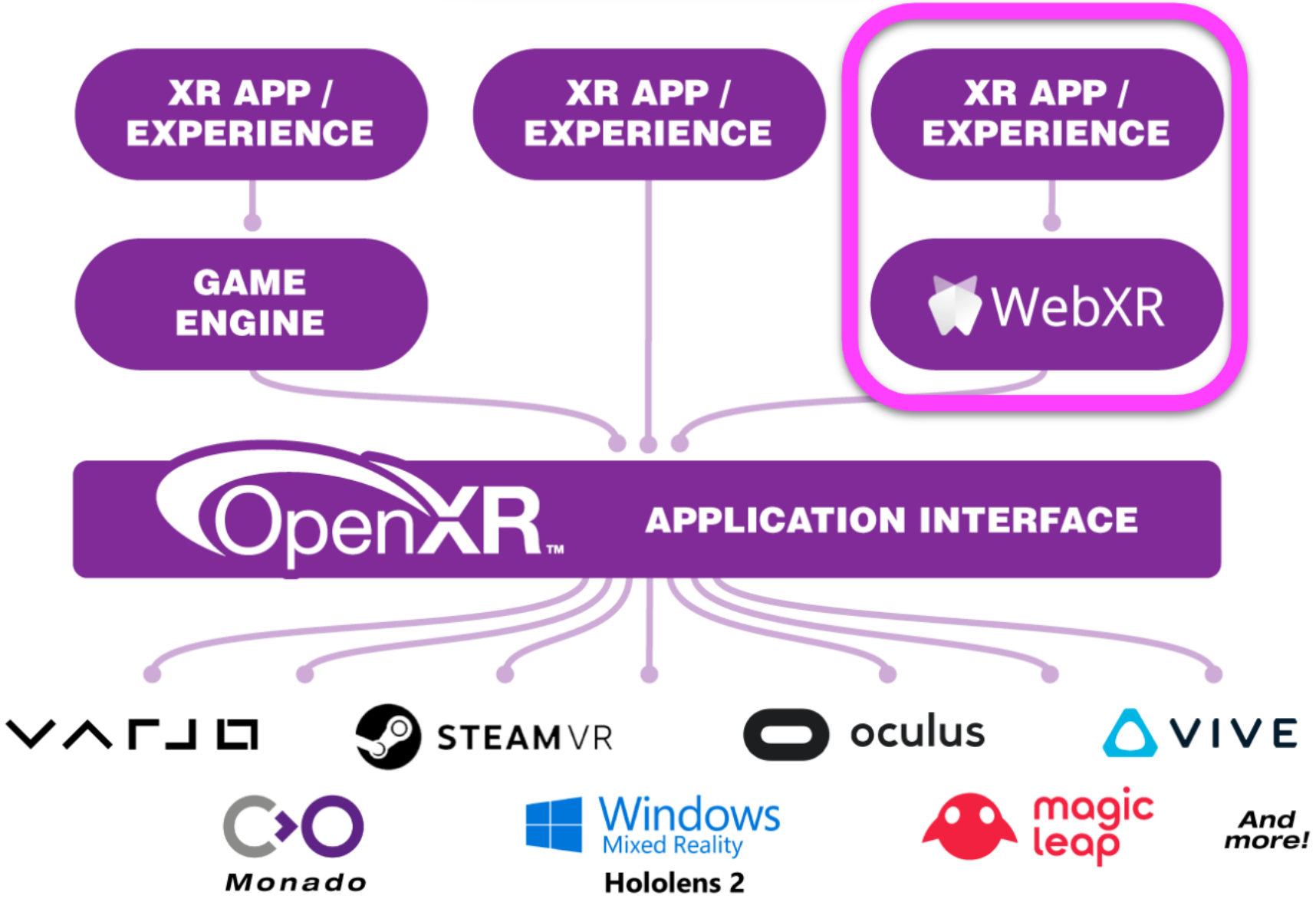
- **Code**: HTML + CSS + JS
  - JS code inside the web page makes WebGL API calls
- **Browser**:
  - browser interprets JS code (using JS Engine)
  - turns WebGL calls into OpenGL calls (binding)
- **OS + Driver**: converts OpenGL calls to
  - DirectX calls on Windows, Metal on Apple (using ANGLE)
  - OpenGL or OpenGL ES calls on other OSes
- **CPU + GPU**: run the **hardware accelerated** code!

Binding example: from JS to C++

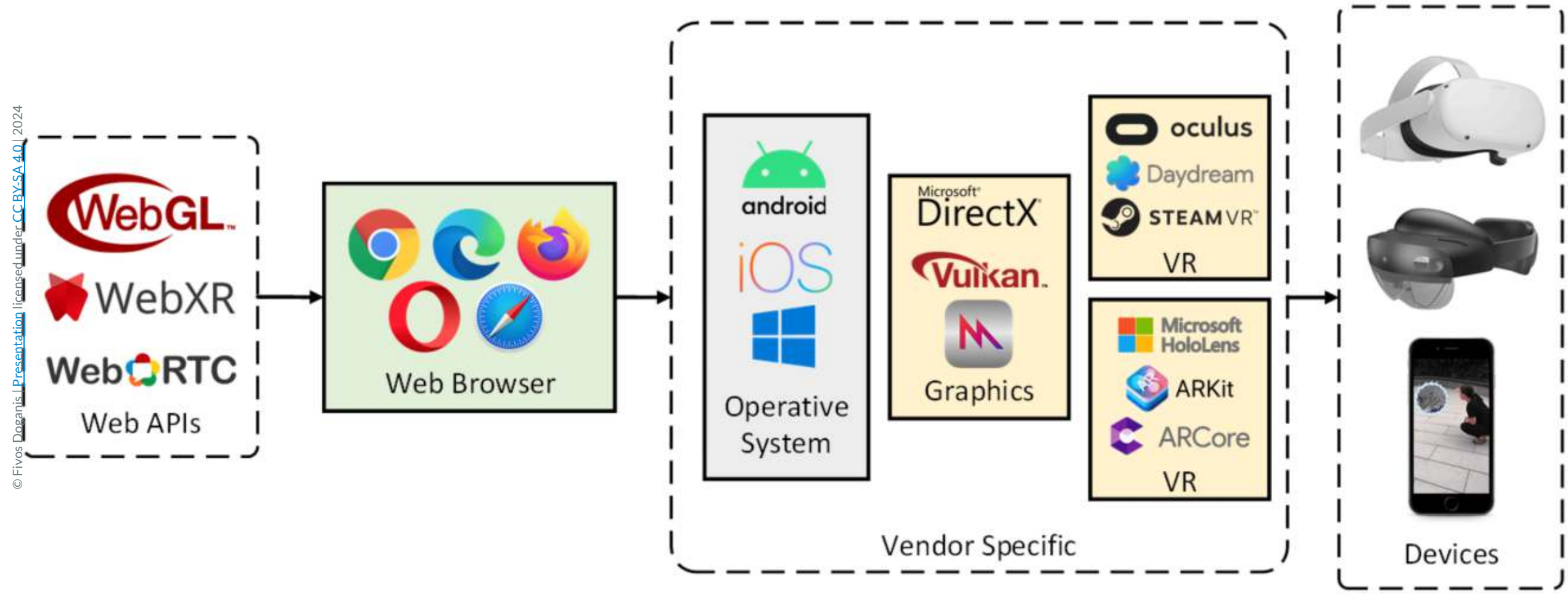
```
gl.drawElements(primitiveType, count, indexType, offset);
```

```
JSValue JSCanvasRenderingContext3D::glDrawElements(JSC::ExecState* exec, JSC::ArgList const& args)
{
    unsigned mode = args.at(0).toInt32(exec);
    unsigned type = args.at(1).toInt32(exec);

    unsigned int count = 0;

    // If the third param is not an object, it is a number, which is the count.
    // In this case if there is a 4th param, it is the offset. If there is no
    // 4th param, the offset is 0
    if (!args.at(2).isObject()) {
        count = args.at(2).toInt32(exec);
        unsigned int offset = (args.size() > 3) ? args.at(3).toInt32(exec) : 0;
        impl()->glDrawElements(mode, count, type, (void*) offset);
    } else {
```

WebXR



OpenXR provides a single cross-platform, high-performance API between applications and all conformant devices.



Hybrid Apps

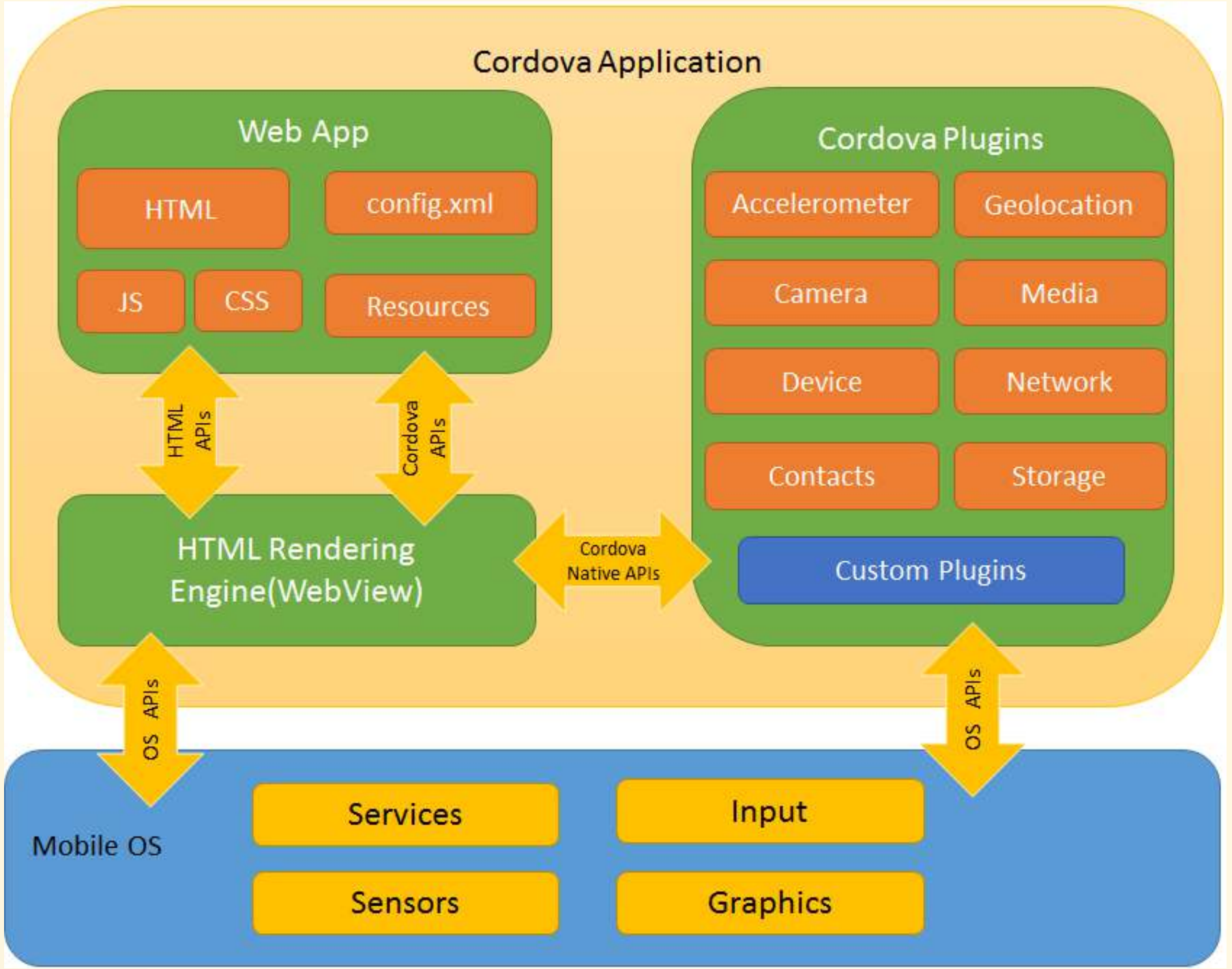
Native + Web

Hybrid Apps

- **Some feature are not available** via JS in the standard browser
  - e.g.: no LiDAR access
- Web Plugins are forbidden
  - security, installation and update issues
- Idea: reuse existing web browser components
  - **native low-level code** (C++, Java, Kotlin, Objective-C, Swift)
  - **create new APIs in JS**, which call the native code
- **issues with non-standard APIs**: if we create our own APIs the code might not be portable

Architecture

- Hybrid App example: ~~PhoneGap~~ **Cordova**
- **NEW**: Ionic Capacitor (WebXR not supported yet)



Why?

- **Pros**
  - **expose missing APIs** in JS: often the only solution!
  - **speed**: call native code, faster than pure JS
  - **portable code**: app logic written in JS
- **Cons**
  - still slower than a 100% native app
  - wasted **performance** when converting between JS and native
  - **lowest common denominator** to satisfy most platforms
  - **non-native look and UI** (not very important for immersive apps)

Setup

How to run the examples locally

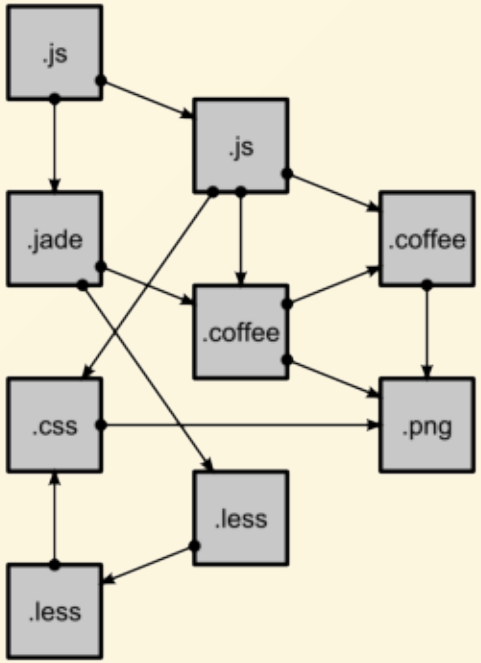
Full setup using NPM

- install Node.js + install npm

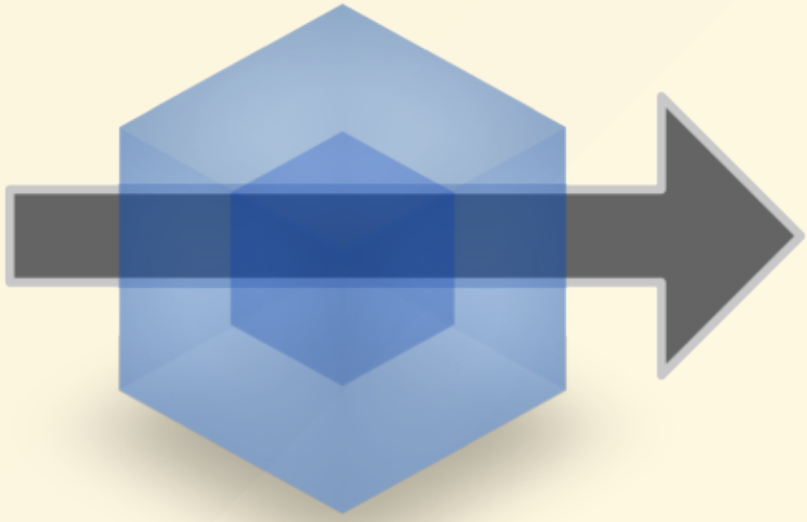
```
sudo apt install nodejs
curl -L https://npmjs.org/install.sh | sudo sh
```

See below 👇

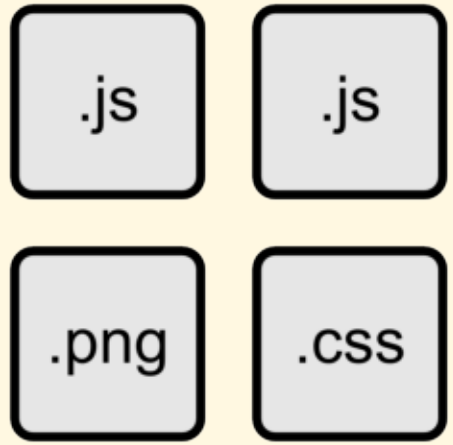
Bundlers



modules
with dependencies



webpack
MODULE BUNDLER



static
assets

Automatic installation

THREE.js + WebXR with "batteries included"

THREE Vite WebXR boilerplate

Preconfigured environment (allows to test all official examples)

https://github.com/fdoganis/three_vite_xr ★

```
git clone https://github.com/fdoganis/three_vite_xr.git

cd three_vite_xr

npm install
```

Run with `npm run dev` or use `F5` in VS Code

Open `http://localhost:5173` in your browser

Tools

- Web development
  - Web browser (Firefox, Chrome, Safari Mobile)
  - Git
  - Code Editor (VSCode)
- Technologies
  - HTML, JS, CSS
  - WebGL, THREE.js
  - WebXR

Install a browser (desktop)

- Firefox installed by default
  - should be enough!
- Chrome
  - to test compatibility and some features
  - alternative: install Chromium on Linux
    - open-source version without proprietary services

```
sudo apt-get install chromium-browser
```

Install Git

```
sudo apt-get install git

git config --global user.name "myusername"
git config --global user.email myname@mymailprovider.com
```

Install VSCode

```
sudo apt update
sudo apt install software-properties-common apt-transport-https wget

wget -q https://packages.microsoft.com/keys/microsoft.asc -O- | sudo apt-key add -

sudo add-apt-repository "deb [arch=amd64] https://packages.microsoft.com/repos/vscode stable main"

sudo apt install code
```

Remove GPG warnings

```
sudo gpgconf --kill dirmngr
sudo chown -R $USER:$USER ~/.gnupg
```

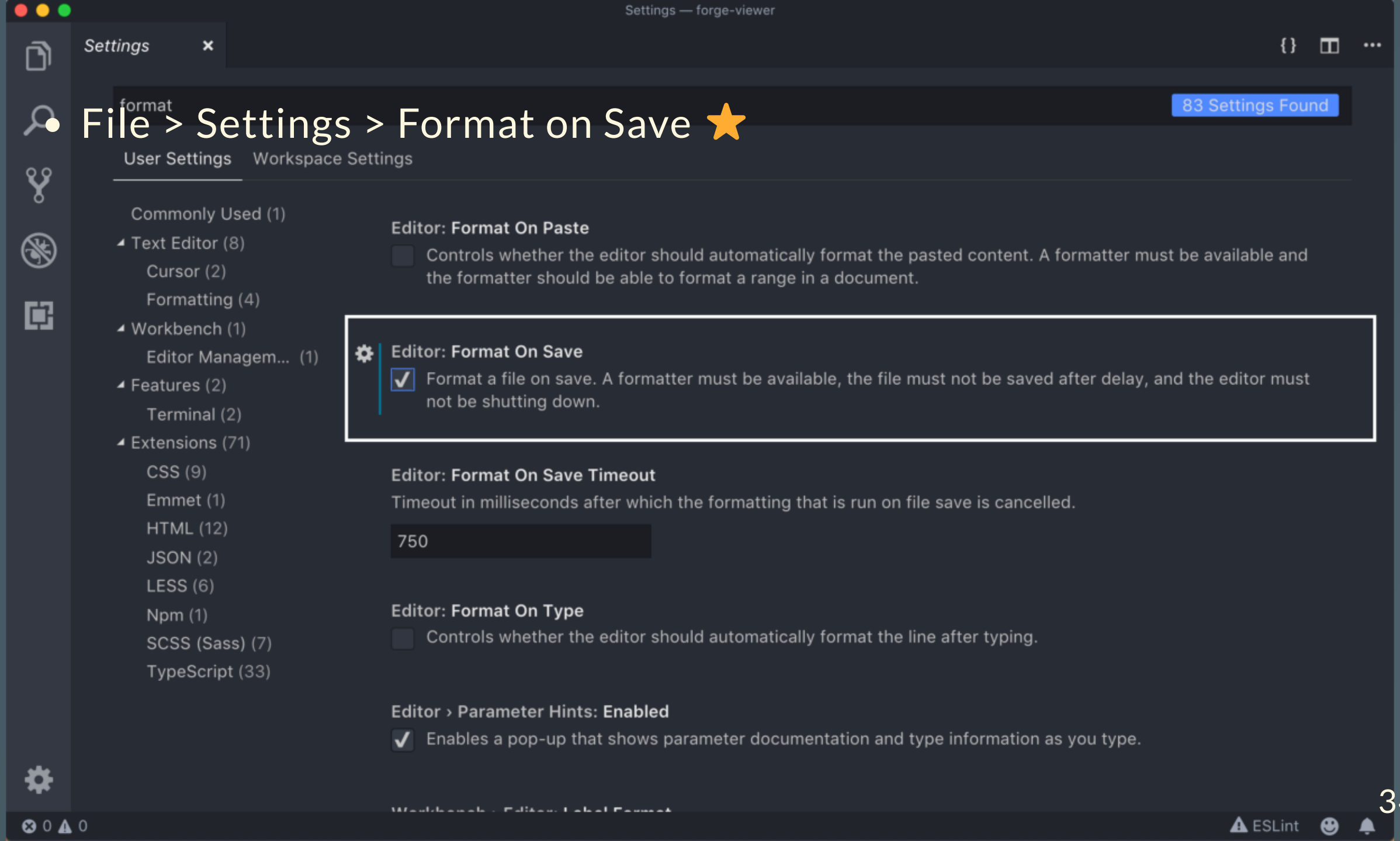
Customize VS Code

- Avoid UI blinking by changing the settings:

```
set window.titleBarStyle to custom
```

- Recommended extensions
  - Live Server
  - Git Graph and/or Git Lens
  - glTF Model Viewer, glTF Tools
  - WebGL GLSL Editor, glsl-canvas
  - Todo Tree, Color Highlight

File > Settings > Format on Save ⭐



Local development caveats

- Impossible to load a file without a user action
- CORS: Cross Origin Resource Sharing
  - one of the many security measures used by web browsers

➡️ need to run a server, like Live Server, or using Python:

```
$ cd /home/somedir
$ python -m SimpleHTTPServer

$ python3 -m http.server
```

Then open `http://localhost:8000` in your browser

Install WebXR browser (mobile)

- **Android** 🤖
  - install the latest mobile **Chrome** version (129+)
- **Meta Quest** ♾️🥽
  - use default **Browser app**
- **Apple Vision Pro** 🍎🥽
  - VR only, no AR 😢
- **iOS** 🍎📱
  - no official WebXR support 😭
  - alternatives below 👇 ~~and here~~ 💀

Install iOS WebXR browser

- ➡️ install XR Browser from the App Store

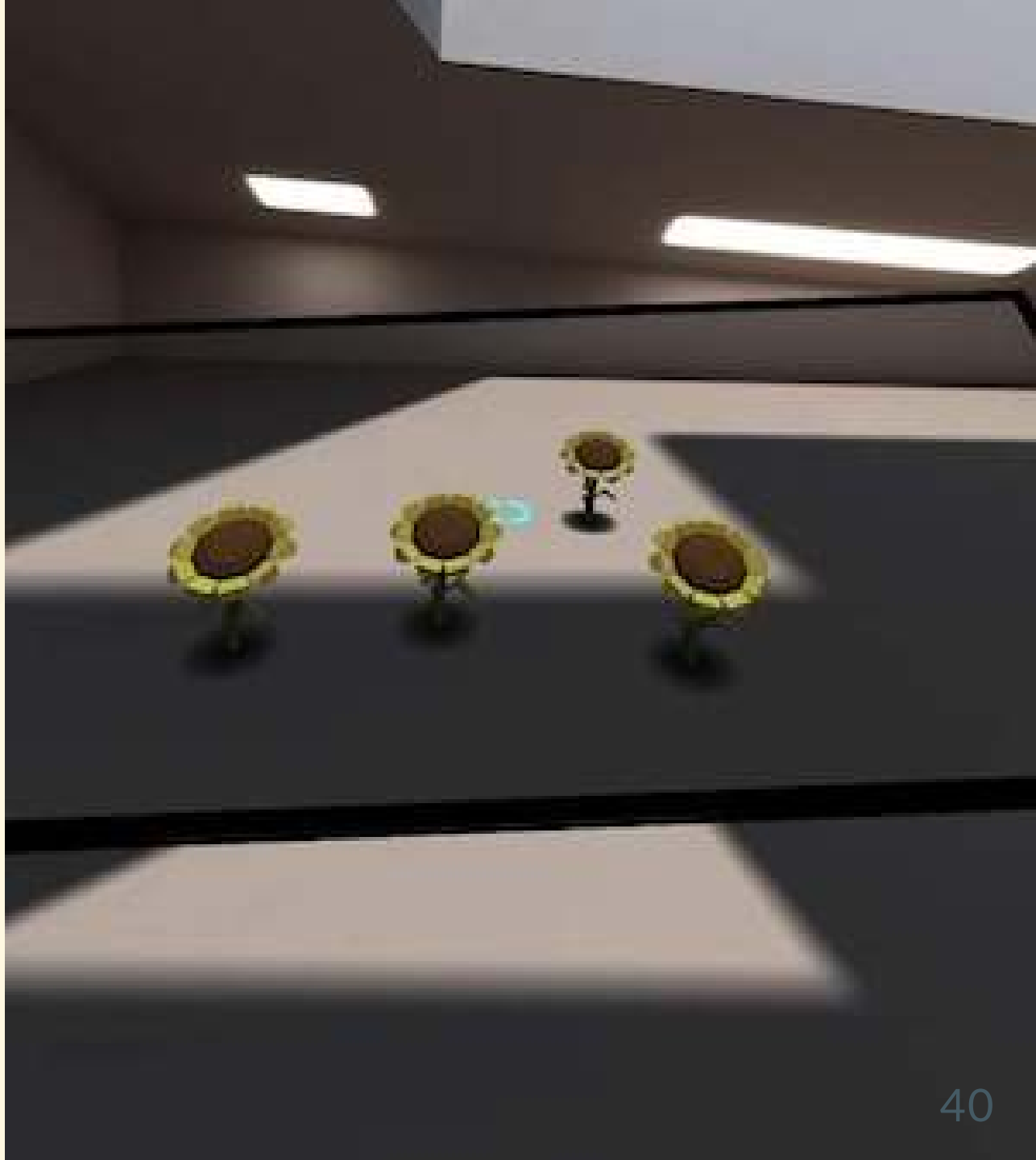
"caniuse"

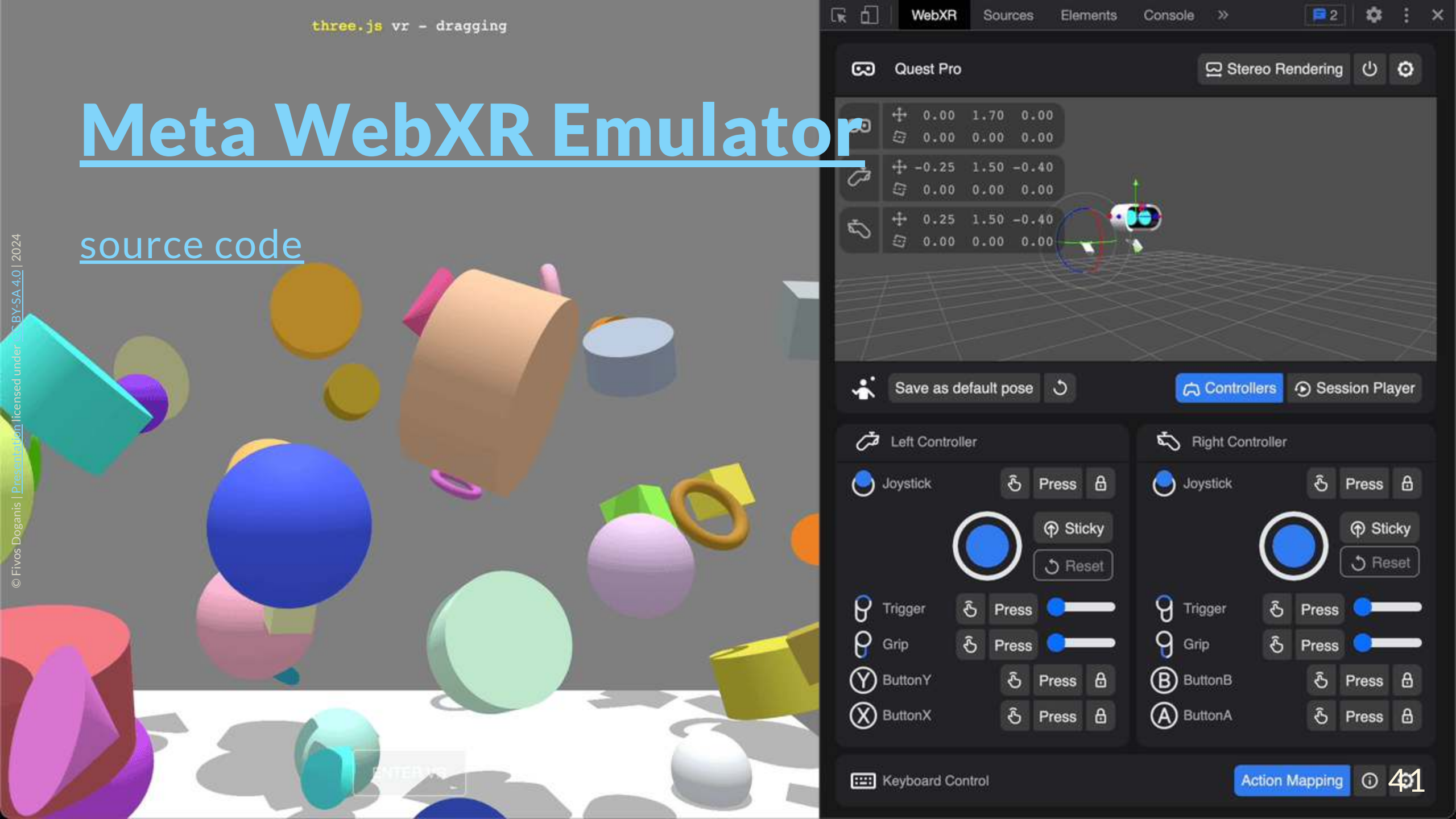
WebXR Report ⭐

Desktop WebXR Emulators

Mozilla WebXR Emulator

- uses WebXR Polyfill
- fake mobile AR device
- very convenient when you don't have an AR device or for debugging
- hand tracking (WIP)
- **NO LONGER ACTIVE** forked by Meta 👇

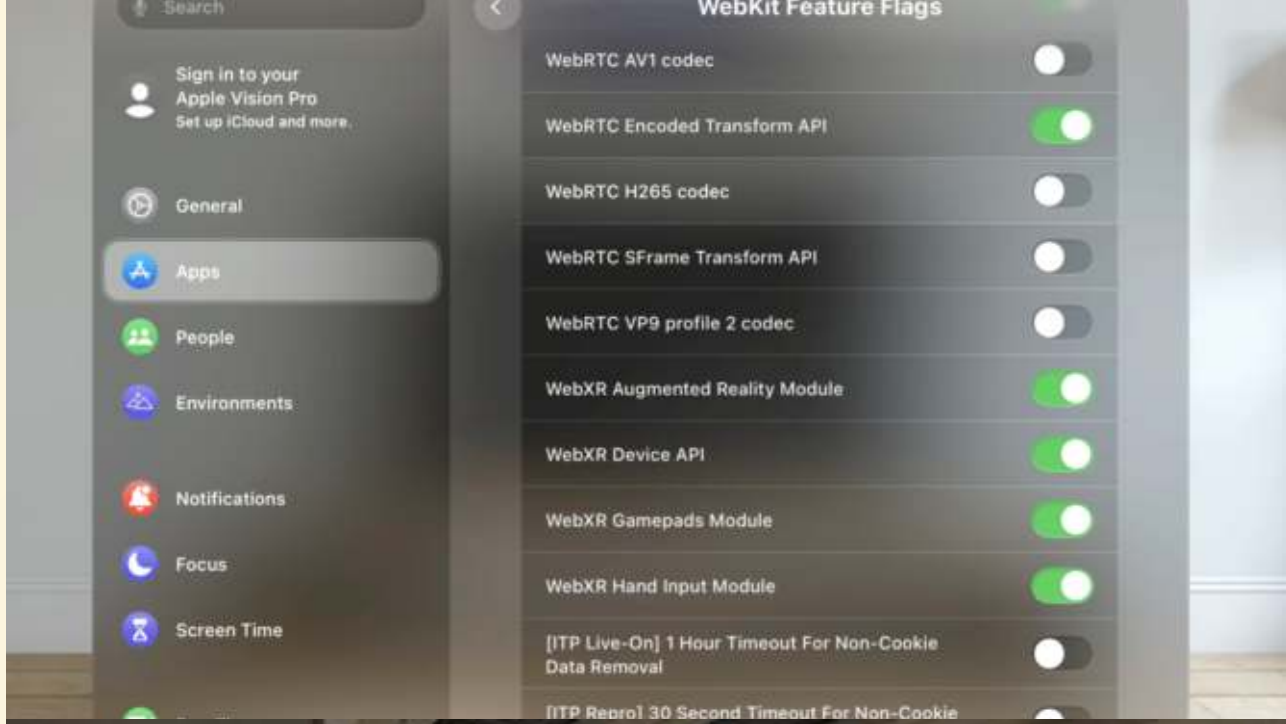




Meta WebXR Emulator

source code

WebXR in Safari inside Apple Vision Pro Emulator



Demo

WebXR Concepts

WebXR Basic Concepts XR + AR ⭐

- **Tracking** (spaces) and **geometry** of the real scene:
  - detect planes and geometry (point cloud or mesh) using SLAM, or similar technologies
- **XR Frame**: RGB image + camera info (pose, focal, tracking, light)
- **Hit test** intersection between a virtual ray and the real scene
  - frequent constraint: RGB camera + depth estimation
- **Anchors** and **worldmap** :
  - points of interest placed by the used
  - updated continuously as the real world gets reconstructed

Advanced WebXR Concepts

- **Occlusion handling**
  - human occlusion (ARKit)
  - real world occlusion (ARKit + LiDAR, ARCore)
- Perception
  - of the environment (Vision, IA, LiDAR)
    - reconstruction + classification floor, wall, table
  - of the user
    - hand gestures, gaze, intentions

Standard APIs

- ~~WebVR~~ ❌ avoid, obsolete
  - but sometimes the only API available
- WebXR ✅ == **"WebVR 2.0 + AR"**
  - ⚠️🚧 W3C Draft API, not stable yet, evolving fast
  - Chrome 81+ , AR still experimental, cf. `chrome://flags`
- **Polyfills**
  - allow converting between APIs
  - let you use the latest WebXR API or the old WebVR
  - allow a limited WebXR API emulation if needed

Code examples : WebXR needs 3D

- WebXR + **WebGL**
  - https://github.com/immersive-web/webxr-samples/blob/master/immersive-ar-session.html
  - https://github.com/immersive-web/webxr-samples/blob/master/hit-test.html
- WebXR + **THREE**
  - https://threejs.org/examples/webxr_ar_hittest
- WebXR + **A-Frame**
  - Basketball: https://ada.is/blog/2021/01/14/making-an-ar-game/

WebXR AR Module

API overview using pseudo-code

Security constraints 🚨

- **permissions**
  - camera, location, movement
- ⚠️ `https` **mandatory**
  - use localhost + SSL, or **glitch**, or github.io, vercel etc.
  - ➡️ better: use **Cloudflare** or install ngrok (see README)
    - allows to create a `https` tunnel very easily
- **requires user action to start**
  - AR / VR / XR Button to switch to AR

AR Initialization

```
isSessionSupported(‘immersive-ar’);

// RequestSession on Button press
navigator.xr.requestSession

// Add listener for ARButton Press

// Request reference spaces
localReferenceSpace = await session.requestReferenceSpace(‘local');
viewerReferenceSpace = await session.requestReferenceSpace(‘viewer');

// Request hitTest

session.requestHitTestSource

// RequestAnimationFrame
// NOTE: THREE.js must use
 renderer.setAnimationLoop
 // instead of window.requestAnimationFrame
// Or else use session.requestAnimationFrame(render)
```

Draw

```
// On each Draw
// Callback on every draw, with an XRFrame

const render = (t, frame) => {

    const pose = frame.getViewerPose(localReferenceSpace);

    frame.getPose(localReferenceSpace, viewerReferenceSpace).transform.matrix

    const hitTestResults = frame.getHitTestResults( hitTestSource );
    const hit = hitTestResults[ 0 ];
    reticle.matrix.fromArray( hit.getPose(viewerReferenceSpace ).transform.matrix );

}
```

Selection (onTouch)

Example:
https://github.com/mrdoob/three.js/blob/master/examples/webxr_ar_cones.html

```
// Get hand, controller, or phone
controller = renderer.xr.getController( 0 );

// See also selectstart, selectend, squeeze etc.
controller.addEventListener( 'select', onSelect );

scene.add( controller );

// Before rendering, update the controller, and apply position to mesh (in meters)
mesh.position.set( 0, 0, - 0.3 ).applyMatrix4( controller.matrixWorld );
```

Let's Code!

WebXR + THREE.js ⭐

- download https://github.com/fdoganis/three_vite_xr
  - see above
- choose a webxr example
  - preferably with `ar /` in its description
  - ths simplest is **AR Cones** : https://threejs.org/examples/webxr_ar_cones
- make it run on your phone (or on an emulator)
  - you can generate a QR code with the URL! 👇

Creating your own QR code

- https://duckduckgo.com
  - "qr " + url
  - ⚠️ you must modifiy the link in THREE.js examples to remove iframes (**iframes are not supported by XR Viewer on iOS**) : (~~https://threejs.org/examples/?q=cones#webxr_ar_cones~~) ➡️ https://threejs.org/examples/webxr_ar_cones
- Alternatives:
  - use Google Chrome's QR code generator
  - https://www.the-qrcode-generator.com/
  - https://codepen.io/chriscoyier/pen/QyPbXz

Build: reminders

- `npm install`
- `npm run build`
- if you have any issues, check the security constraints

Replace the cone with another model

- SketchFab ⭐
  - A-Frame component
  - Downloaded model
- ~~Google Poly~~ 💀 ➡️ poly.pizza
  - example
  - A-Frame component
  - Downloaded

Add XR to your THREE.js project! ⭐

Port your THREE.js project to use the real world!

- start with THREE.js' `webxr_ar_cones` example above
- replace the cone creation with your solar system
  - create the solar system only once
  - change its position if it has already been created

Challenges 💪

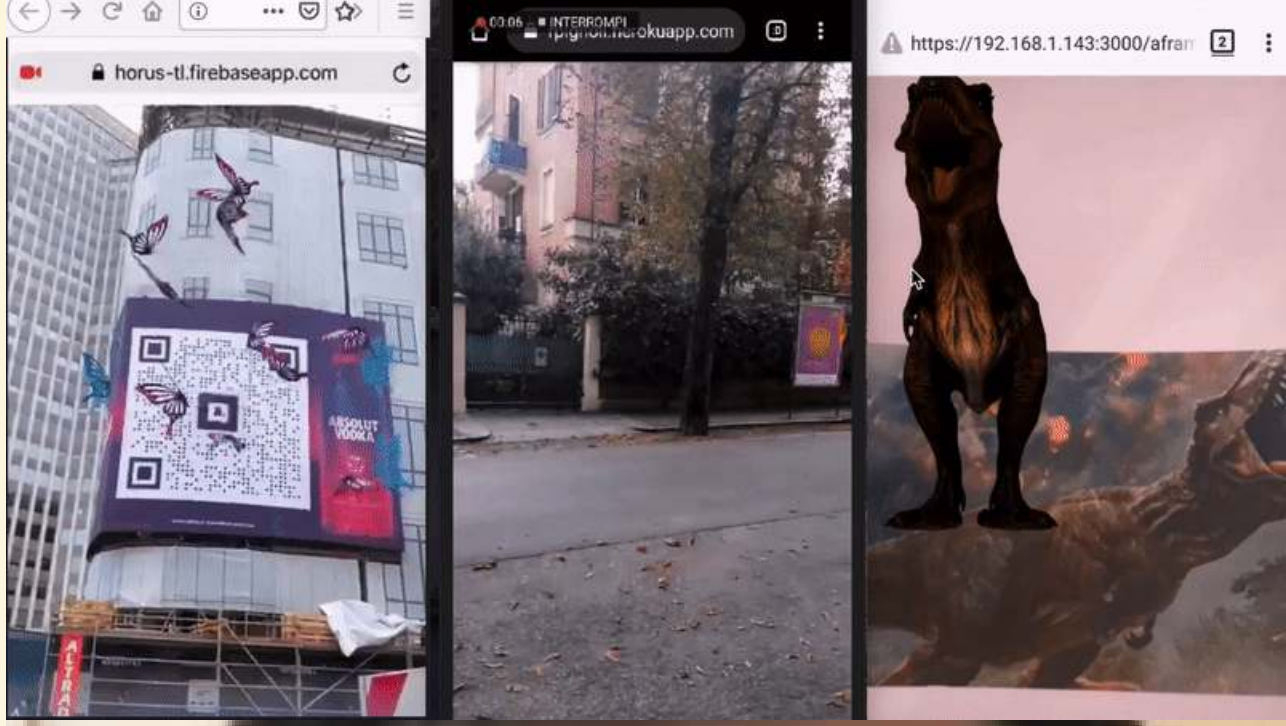
- 1️⃣ place the solar system on top of a horizontal plane
  - use a hit test
  - see https://threejs.org/examples/?#webxr_ar_hittest
- 2️⃣ select a sphere and move it
  - use `attach` while dragging it
  - see https://threejs.org/examples/#webxr_xr_dragging
- 3️⃣ interpolate between the two positions
  - `new TWEEN.Tween(obj.position).to({x:, y:, z:}, 500).start()`
  - see this TWEEN.js example

Alternative AR Libraries

- ~~three.ar.js~~, ~~aframe-ar~~: obsolete, avoid
- JeelizAR ★
  - lightweight fast (deep learning for object and face detection)
- MediaPipe, Handsfree.js encapsulates JeelizAR and TensorFlow.js
- ~~awe.js~~: not free
- ~~8th Wall~~
  - not free, proprietary and very restrictive license, **avoid**!
  - does not allow evaluation if you are a developer!
- ~~Wikitude~~: not free, n web

More libraries

- **AR.js**
  - THREE.js + marker
  - THREE.js + NFT
  - **locar.js** : THREE.js + GPS
- **MindAR**
  - simple **image tracking** with THREE.js
  - **face tracking** with THREE.js

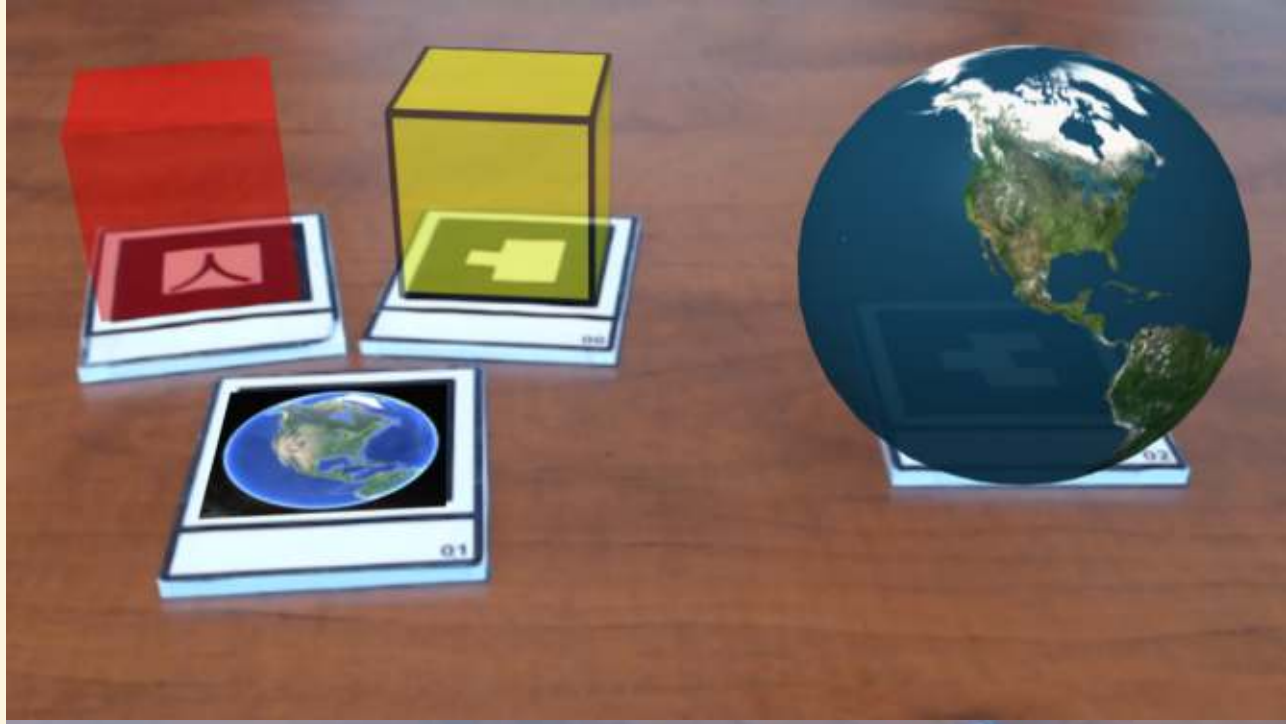


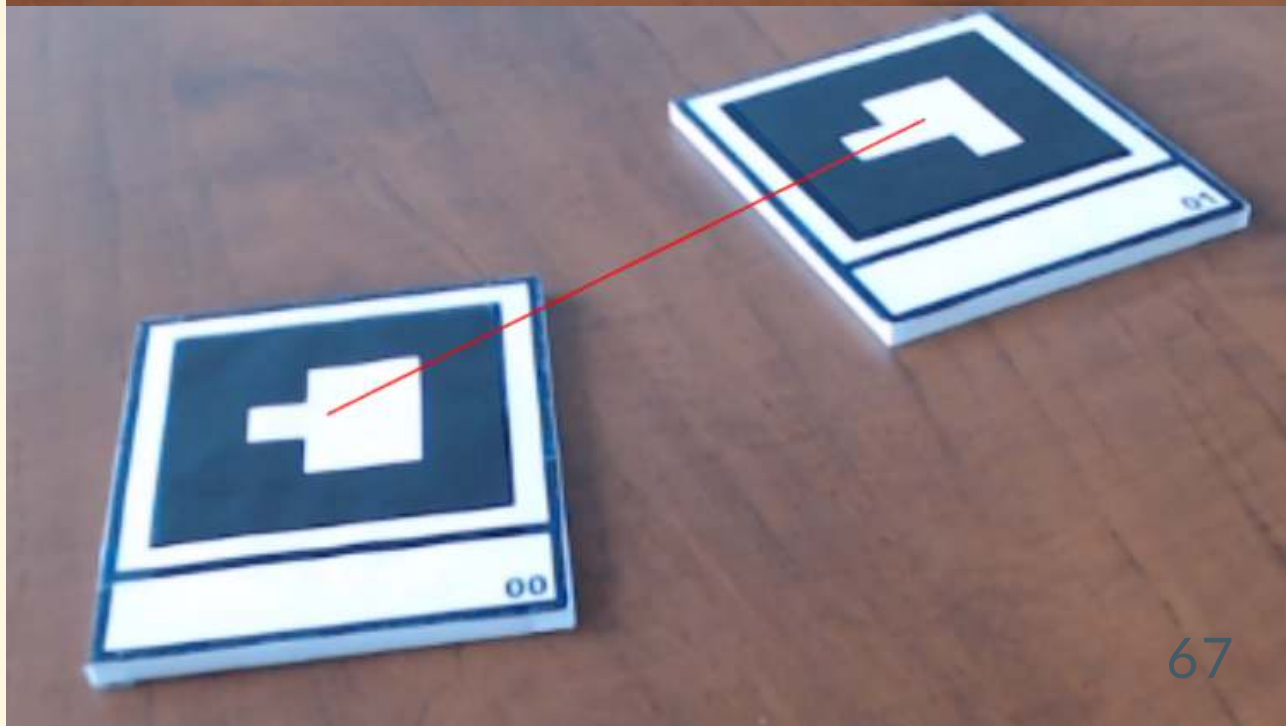
App examples

- Many (old) examples using **AR.js + THREE.js** + THREEx
  - official
    - basic example
    - you can choose between live video, simple photo and a video a file!
  - Lee Stemkoski

More examples

- Official A-Frame examples
- Lee Stemkoski
  - A-Frame
  - **A-Frame + AR.js**
- FrameVR
  - course for beginners
  - inspiration for your projects





Computer Vision

- OpenCV.js : official JS port (wasm via emscripten)
  - terrible documentation but nice tutorials, see this Mozilla article
- JSFeat: lightweight 100% js library (old code)
- Tracking.js: computer vision lib (old code)
- headtrackr, PicoJS: face tracking
- ARuCo Marker tracking: ARToolkit alternative
- Tensorflow.js: recent Vision + IA demos
  - PoseNet
  - HandPose 3D

Extras

Courses

https://medium.com/sopra-steria-norge/get-started-with-augmented-reality-on-the-web-using-three-js-and-webxr-part-1-8b07757fc23a ★

https://codelabs.developers.google.com/ar-with-webxr#0

https://medium.com/arjs/webar-playground-ar-in-a-few-clicks-67a08cfb1534

https://blog.halolabs.io/building-ar-vr-with-javascript-and-html-97af4434bcf6

Guides

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

Code links : Browser-based AR and VR

http://webglworkshop.com/presentations/Workshop31-ar-vr.html#/21

https://webxr.io/webar-playground/

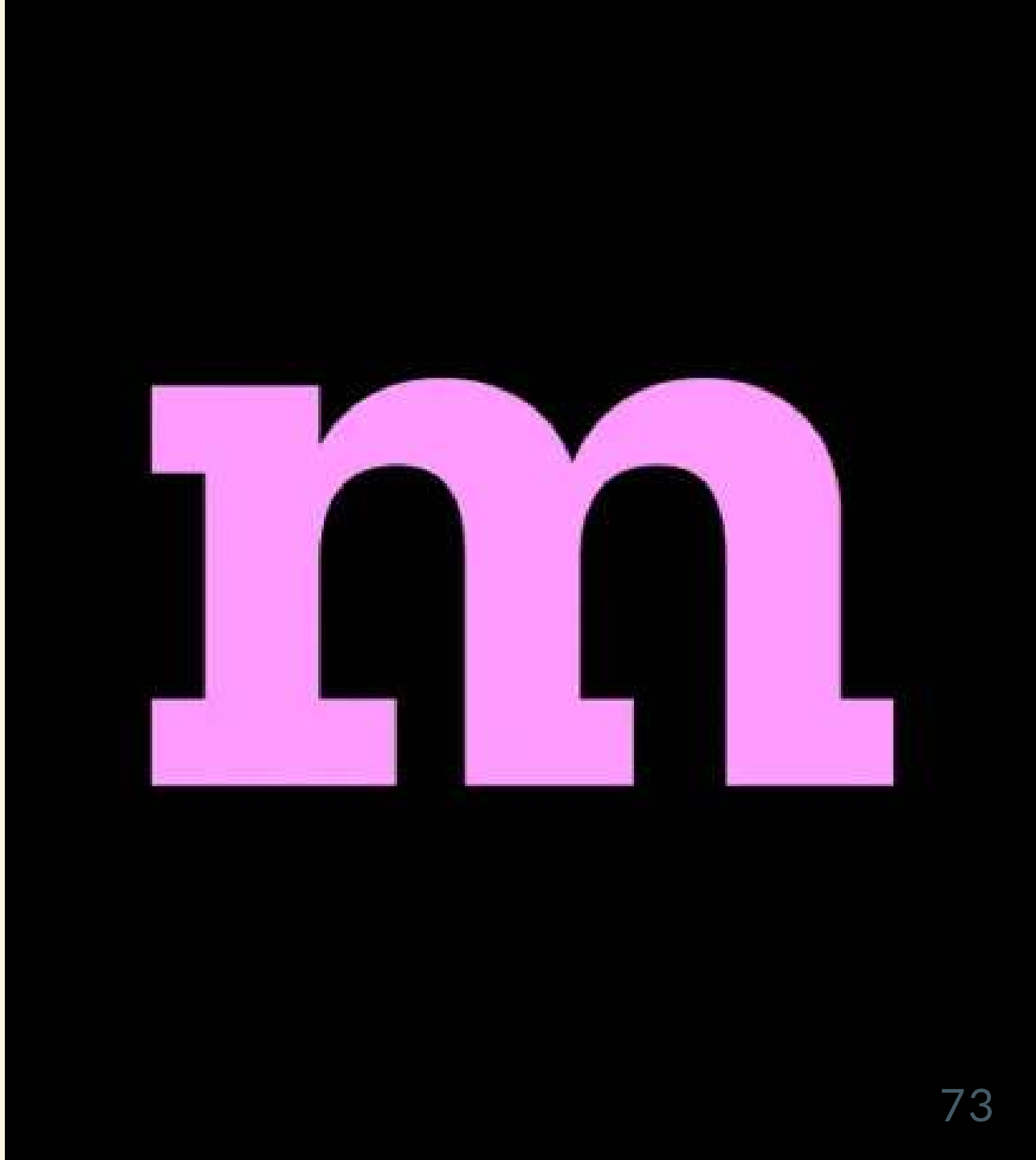
http://learningthreejs.com/blog/2015/07/16/hatsune-miku-dancing-in-augmented-reality/

http://studioknol.com/phase-two-building-with-virtuality/

https://github.com/rodrigocam/ar-gif
https://github.com/XingMeansOK/slamjs_samples (RGBD)

Mozilla's (deprecated) XR Viewer (1)

- Install Mozilla's WebXR Viewer from the App Store



Mozilla's (deprecated) XR Viewer (2)

- **DO NOT use** URLs with **iframes**!

For example:

https://threejs.org/examples/?q=cones#webxr_ar_cones

will NOT work (you will get a confusing "WebXR not available" message on your XR Button)

Use this URL instead:

https://threejs.org/examples/webxr_ar_cones

Mozilla's (deprecated) XR Viewer (3)

- **DO NOT use Dark Mode**, otherwise all the colors in your 3D **graphics will look inverted**
  - you can **force XRViewer to always use the light theme**
    - check the in-app **settings**

Mozilla's (deprecated) XR Viewer (4)

- Remember to **clear your cache** frequently if you are developing an app and you are not seeing your code changes
  - check **Data Management** from the in-app **settings**

Mozilla's (deprecated) XR Viewer (5)

- the site https://webxr-ios.webxrexperiments.com/ seems to be down.
  Symptom: "Start AR" has no effect.
- use updated WebXR polyfill
- ➡️ provide your own polyfill in the general settings:

Settings / XRViewer / WebXR Polyfill URL :
https://arenaxr.org/webxrios.js

URL copied from Anthony Rowes' "XR Browser" (maintained!) app