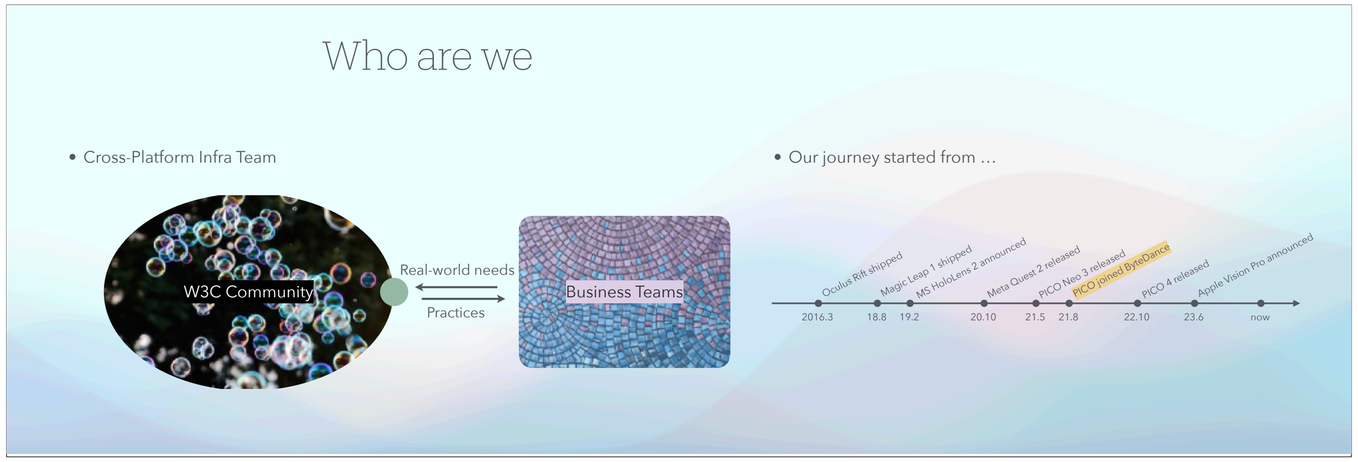
Practicing Immersive Web APIs

Yifei Yu (ByteDance). yuyifei.r@bytedance.com 2024.5

This talk shares our exploration on the api, implementation, and practical use of immersive web apis.



Who are we

- Cross-Platform Infra Team
- Our journey started from …

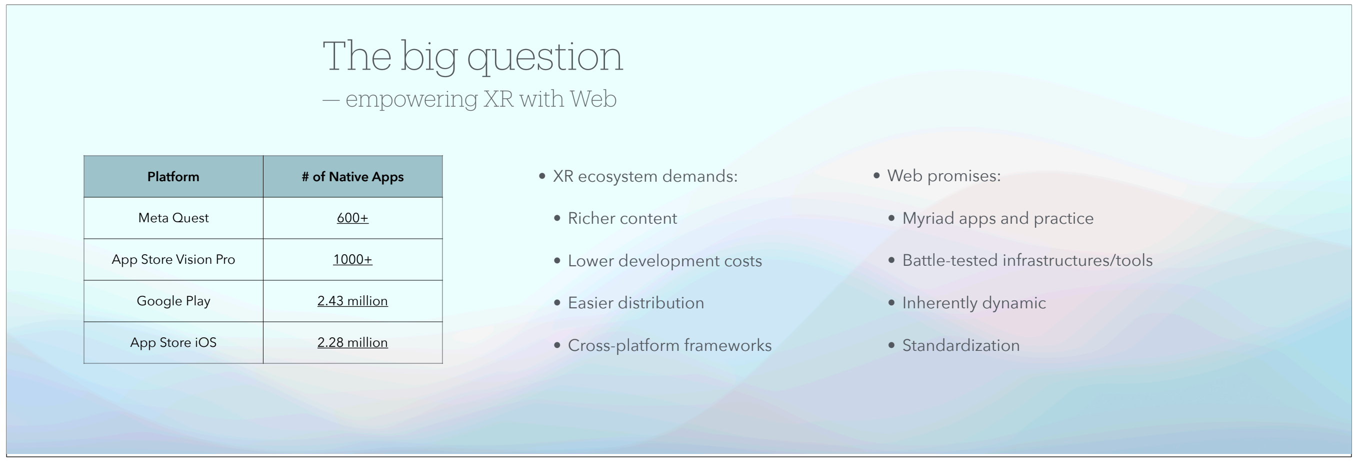
Cross-platform team of ByteDance.

Among the various roles in the community, the (small) green dot is us!
We gain knowledge from the community, but lean towards real world businesses.

We are:
- Practitioner in Web community: w/ exploration and experimentation on new territories.
- Enabler for business teams: w/ latest Web standards and Web engine expertise

We gain ideas from the community, sometimes come up with ours, in return we tell our stories!

21.8 is the time we start working on XR (I'm sure I missed some important dates!), our solutions are available on PICO Neo 3 and PICO 4.



The big question

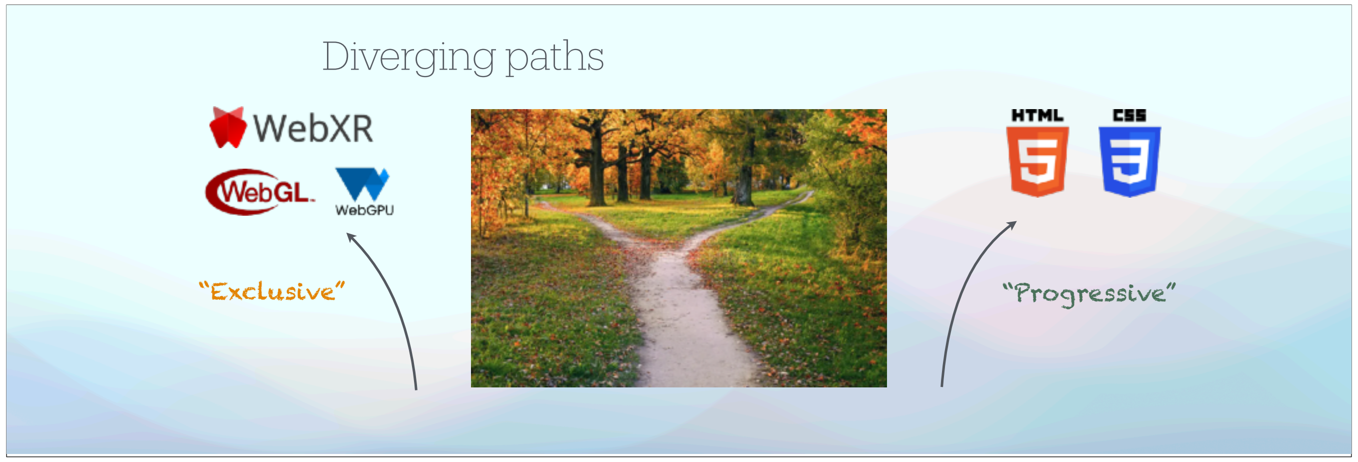
— empowering XR with Web

Platform	# of Native Apps
Meta Quest	600+
App Store Vision Pro	1000+
Google Play	2.43 million
App Store iOS	2.28 million

- XR ecosystem demands:
  - Richer content
  - Lower development costs
  - Easier distribution
  - Cross-platform frameworks
- Web promises:
  - Myriad apps and practice
  - Battle-tested infrastructures/tools
  - Inherently dynamic
  - Standardization

XR ecosystem is developing, and web can help!

Quest 600+ https://www.meta.com/experiences/section/1888816384764129/
Google Play 2.43m https://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store/
iOS 2.28m https://www.businessofapps.com/data/app-stores/
Vision Pro 1000+ https://www.roadtovr.com/apple-vision-pro-1000-apps/



Diverging paths

WebXR

WebGL WebGPU

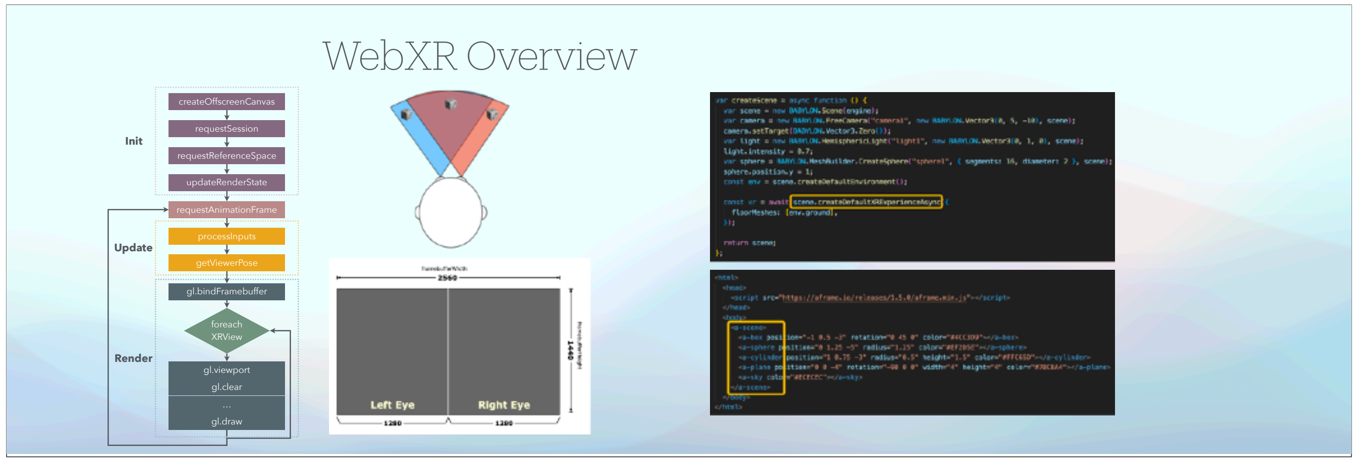
“Exclusive”

HTML CSS

“Progressive”

Web on XR currently has 2 paths:

- WebXR which renders the whole scene (exclusive)
- Progressively adding 3D-capabilities onto HTML/CSS, also progressively adding Web content onto existing 3D XR content.

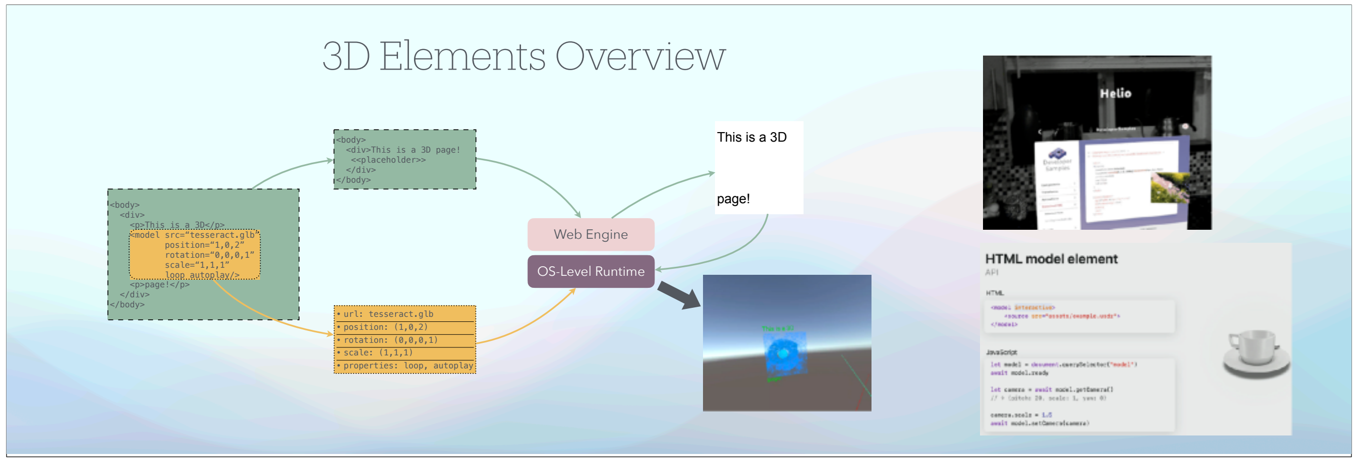


```
var createScene = async function () {
  var scene = new BABYLON.Scene(engine);
  var camera = new BABYLON.FreeCamera("camera1", new BABYLON.Vector3(0, 5, -10), scene);
  camera.setTarget(BABYLON.Vector3.Zero());
  var light = new BABYLON.HemisphericLight("light1", new BABYLON.Vector3(0, 1, 0), scene);
  light.intensity = 0.7;
  var sphere = BABYLON.MeshBuilder.CreateSphere("sphere1", { segments: 16, diameter: 2 }, scene);
  sphere.position.y = 1;
  const env = scene.createDefaultEnvironment();

  const xr = await scene.createDefaultXRExperienceAsync({
    floorMeshes: [env.ground],
  });

  return scene;
};
```

```
<html>
  <head>
    <script src="https://aframe.io/releases/1.5.0/aframe.min.js"></script>
  </head>
  <body>
    <a-scene>
      <a-box position="-1 0.5 -3" rotation="0 45 0" color="#4CC3D9"></a-box>
      <a-sphere position="0 1.25 -5" radius="1.25" color="#EF2D5E"></a-sphere>
      <a-cylinder position="1 0.75 -3" radius="0.5" height="1.5" color="#FFC65D"></a-cylinder>
      <a-plane position="0 0 -4" rotation="-90 0 0" width="4" height="4" color="#7BC8A4"></a-plane>
      <a-sky color="#ECECEC"></a-sky>
    </a-scene>
  </body>
</html>
```

Basic overview of WebXR, explaining how it renders to the eye buffers and cannot coexist with other content.

Web ecosystem has good abstractions (frameworks) above WebXR, making it easy to adopt.



3D Elements Overview

```
<body>
  <div>
    <p>This is a 3D</p>
    <model src="tesseract.glb"
           position="1,0,2"
           rotation="0,0,0,1"
           scale="1,1,1"
           loop autoplay/>
    <p>page!</p>
  </div>
</body>
```

```
<body>
  <div>This is a 3D page!
   <<placeholder>>
  </div>
</body>
```

```
• url: tesseract.glb
• position: (1,0,2)
• rotation: (0,0,0,1)
• scale: (1,1,1)
• properties: loop, autoplay
```

HTML model element

API

HTML

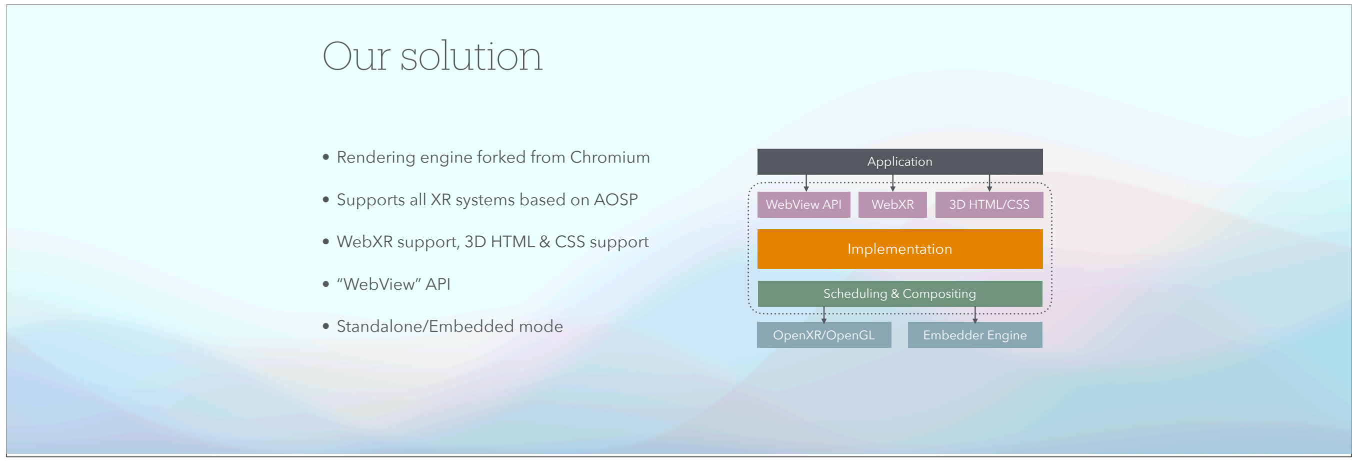
```
<model interactive>
    <source src="assets/example.usdz">
</model>
```

JavaScript

```
let model = document.querySelector("model")
await model.ready

let camera = await model.getCamera()
// → {pitch: 20, scale: 1, yaw: 0}

camera.scale = 1.5
await model.setCamera(camera)
```

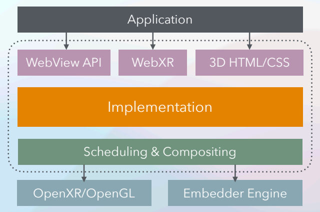
Basic overview of the progressive path. Web engines does not render 3D content by itself. Instead it delegates to a lower level runtime, which composite everything in the 3D space.

Example: Magic Leap 1 and Safari on Vision Pro has declarative apis for Web to render in 3D, each on top of there system level engine (Lumin/RealityKit).



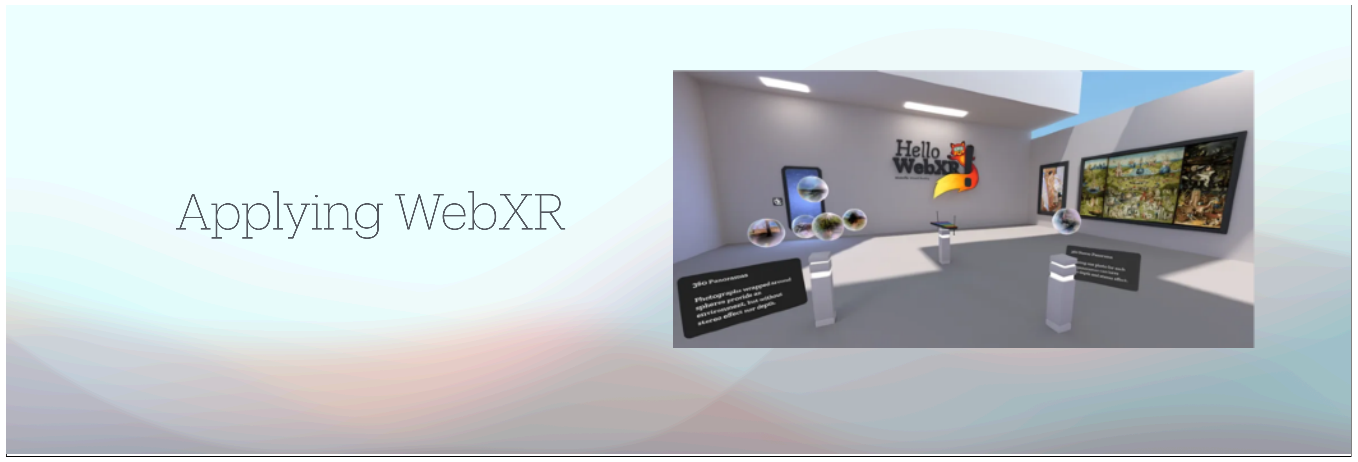
Our solution

- Rendering engine forked from Chromium
- Supports all XR systems based on AOSP
- WebXR support, 3D HTML & CSS support
- "WebView" API
- Standalone/Embedded mode



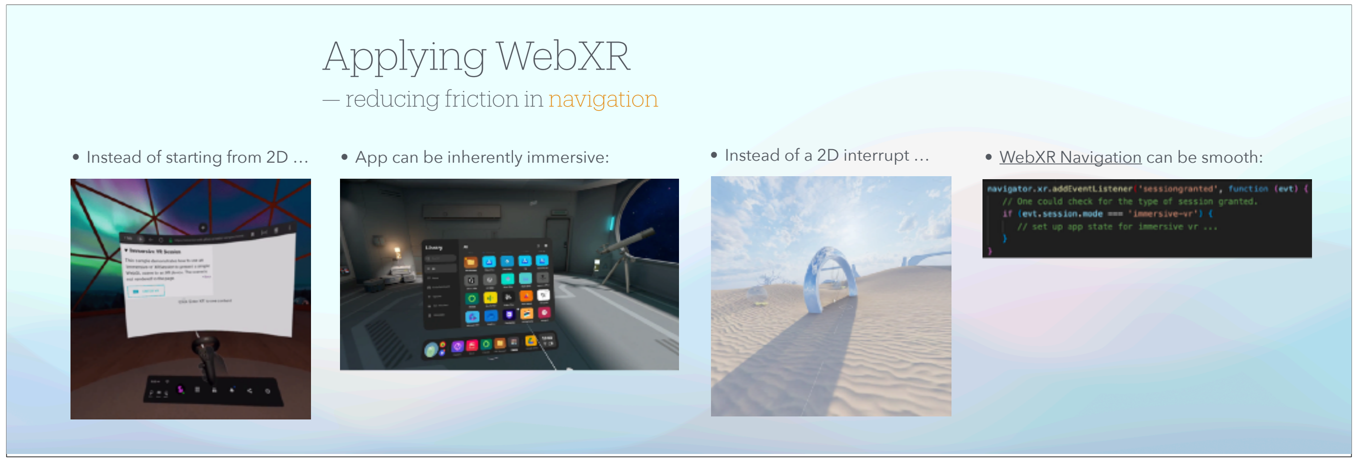
Our solution.

Standalone: basically for WebXR. Packages an Web app into an android app.
Embedded: integrate with 3D Engine or another runtime.



Applying WebXR

Hello WebXR!
360 Panoramas
Photographs wrapped around spheres provide an environment, but without stereo effect nor depth.



Applying WebXR

— reducing friction in navigation

- Instead of starting from 2D …
- App can be inherently immersive:
- Instead of a 2D interrupt …
- WebXR Navigation can be smooth:

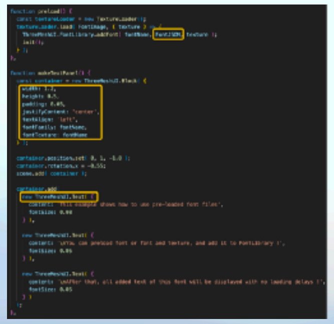
```
navigator.xr.addEventListener('sessiongranted', function (evt) {
   // One could check for the type of session granted.
   if (evt.session.mode === 'immersive-vr') {
      // set up app state for immersive vr ...
   }
}
```

WebXR needs to start from 2D page. When packaged with our solution (or, on PICO, manifested as an “immersive PWA”), can request immersive session with code.

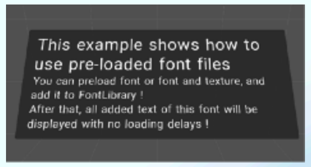
Open another WebXR page also start from 2D. WebXR Navigation implemented.

Applying WebXR

— reducing friction in 2D rendering

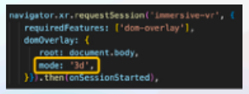


```
function preload() {
  const textureLoader = new TextureLoader();
  textureLoader.load( FontImage, ( texture ) => {
    ThreeMeshUI.FontLibrary.addFont( fontName, FontJSON, texture );
    init();
  } );
};

function makeTextPanel() {
  const container = new ThreeMeshUI.Block( {
    width: 1.2,
    height: 0.5,
    padding: 0.05,
    justifyContent: 'center',
    textAlign: 'left',
    fontFamily: fontName,
    fontTexture: fontName
  } );

  container.position.set( 0, 1, -1.8 );
  container.rotation.x = -0.55;
  scene.add( container );

  container.add(
    new ThreeMeshUI.Text( {
      content: 'This example shows how to use pre-loaded font files',
      fontSize: 0.08
    } ),

    new ThreeMeshUI.Text( {
      content: '\nYou can preload font or font and texture, and add it to FontLibrary !',
      fontSize: 0.05
    } ),

    new ThreeMeshUI.Text( {
      content: '\nAfter that, all added text of this font will be displayed with no loading delays !',
      fontSize: 0.05
    } )
  );
};
```


This example shows how to use pre-loaded font files
You can preload font or font and texture, and add it to FontLibrary !
After that, all added text of this font will be displayed with no loading delays !

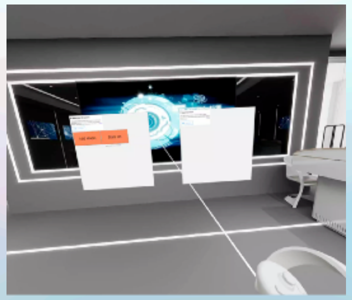
- Converts to Canvas API calls under the hood
- Handles Fonts & Layout on JS side
- Impact on perf, code size, rendering quality
- How about good old HTML & CSS?

- "3D" DOM Overlay:

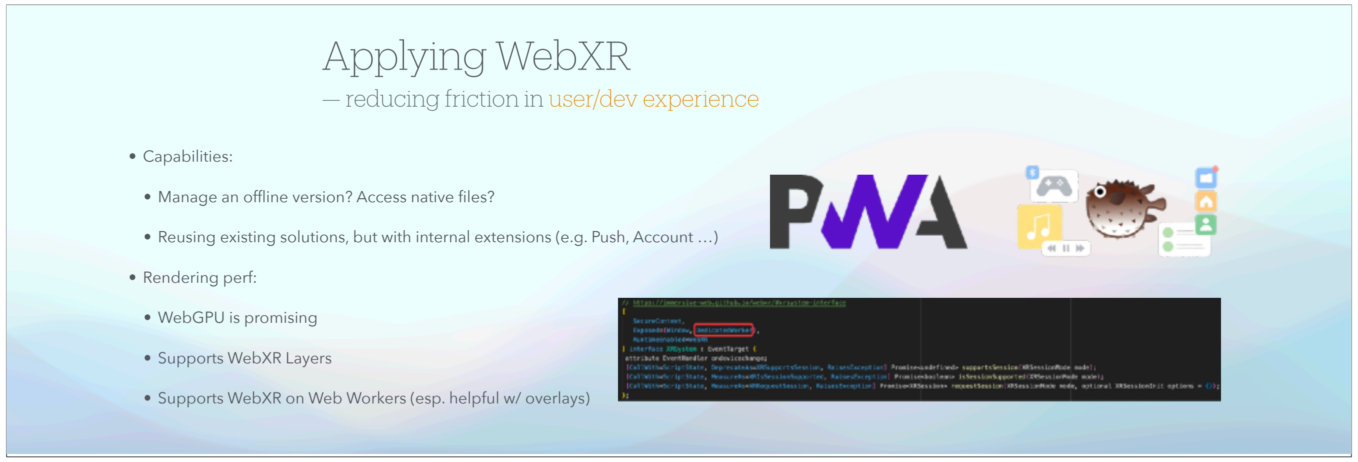


```
navigator.xr.requestSession('immersive-vr', {
    requiredFeatures: ['dom-overlay'],
    domOverlay: {
      root: document.body,
      mode: '3d',
  }}).then(onSessionStarted),
```

- ATW using WebXR depth buffer
- Care needed for UX



Rendering 2D content in WebXR is not good developer experience. Instead we extend WebXR DOM Overlay to 3D.



Applying WebXR

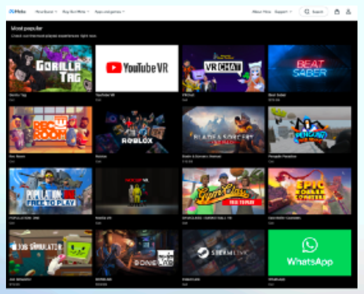
— reducing friction in user/dev experience

- Capabilities:
  - Manage an offline version? Access native files?
  - Reusing existing solutions, but with internal extensions (e.g. Push, Account …)
- Rendering perf:
  - WebGPU is promising
  - Supports WebXR Layers
  - Supports WebXR on Web Workers (esp. helpful w/ overlays)

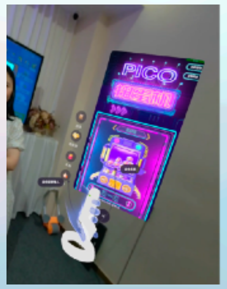
```
// https://immersive-web.github.io/webxr/#xrsystem-interface
[
  SecureContext,
  Exposed=(Window, DedicatedWorker),
  RuntimeEnabled=WebXR
] interface XRSystem : EventTarget {
 attribute EventHandler ondevicechange;
 [CallWith=ScriptState, DeprecateAs=XRSupportsSession, RaisesException] Promise<undefined> supportsSession(XRSessionMode mode);
 [CallWith=ScriptState, MeasureAs=XRIsSessionSupported, RaisesException] Promise<boolean> isSessionSupported(XRSessionMode mode);
 [CallWith=ScriptState, MeasureAs=XRRequestSession, RaisesException] Promise<XRSession> requestSession(XRSessionMode mode, optional XRSessionInit options = {});
};
```

We use latest Web capabilities to address developer needs. We implement WebXR Layers and WebXR on Workers for perf.

The status quo and the big transition



- In the long run, Web shines in the general use cases …
- But it can also shine today! In use cases that:
  - Have tight schedule
  - Change frequently
  - Special events, A/B tests …



The progressive path can shine in:
- multitasking use cases
- dynamic part in immersive apps

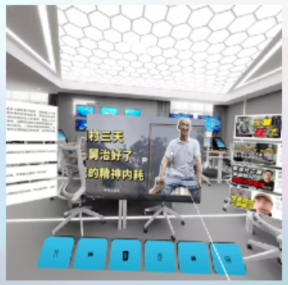
Screenshot shows a 2D event page in a 3D scene, which can be improved with more 3D content.

Turning Hybrid

Detached CSS

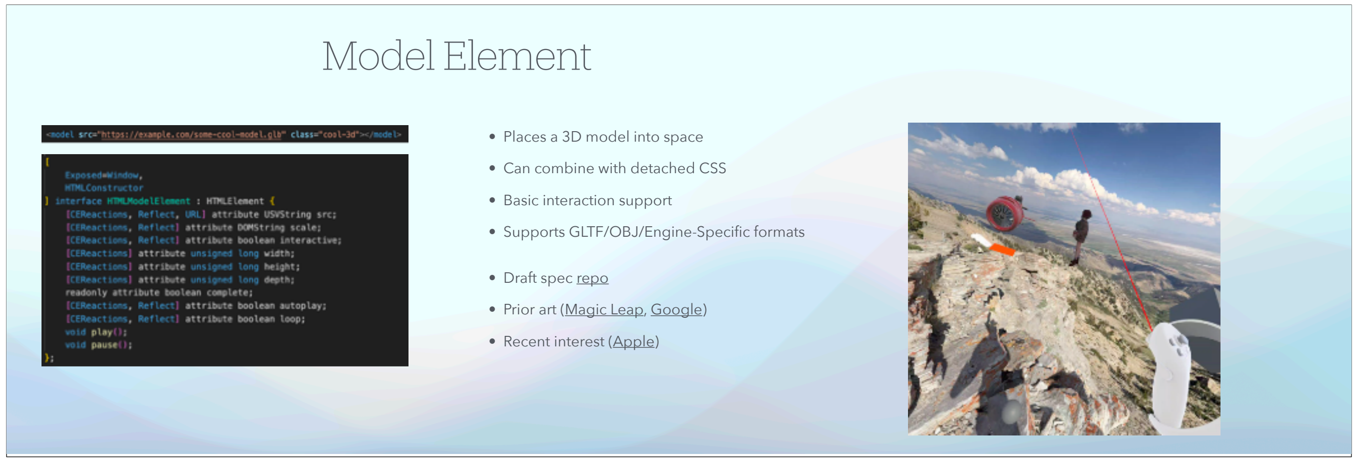
```css
.cool-3d {
  transform-style: detached;
  transform: translate3d(100cm, 50cm, 100cm);
}
```

- Creates an plane detached from the main page
- 3D transforms & animations work intuitively
- Draft spec repo
- Prior art (Magic Leap)



We implemented what is called Detached CSS (originally by Magic Leap).

Video shows a page comprising detached planes.



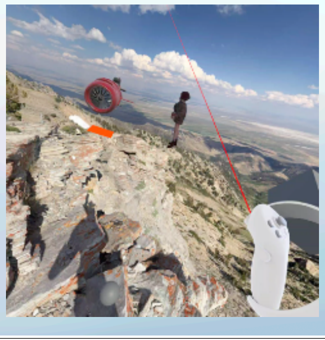
Model Element

```
<model src="https://example.com/some-cool-model.glb" class="cool-3d"></model>
```

```
[
    Exposed=Window,
    HTMLConstructor
] interface HTMLModelElement : HTMLElement {
    [CEReactions, Reflect, URL] attribute USVString src;
    [CEReactions, Reflect] attribute DOMString scale;
    [CEReactions, Reflect] attribute boolean interactive;
    [CEReactions] attribute unsigned long width;
    [CEReactions] attribute unsigned long height;
    [CEReactions] attribute unsigned long depth;
    readonly attribute boolean complete;
    [CEReactions, Reflect] attribute boolean autoplay;
    [CEReactions, Reflect] attribute boolean loop;
    void play();
    void pause();
};
```

- Places a 3D model into space
- Can combine with detached CSS
- Basic interaction support
- Supports GLTF/OBJ/Engine-Specific formats

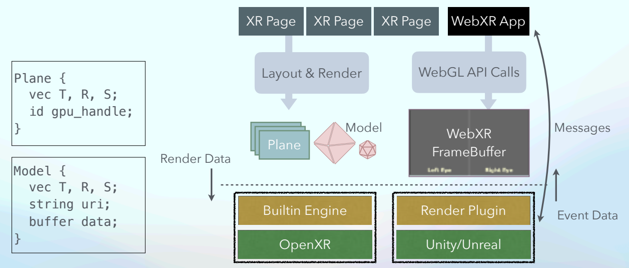
- Draft spec repo
- Prior art (Magic Leap, Google)
- Recent interest (Apple)



Also implemented model element.

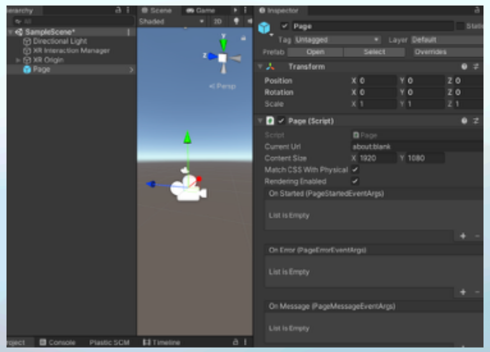
Video shows 3 models in the scene: rocket, girl, and skybox.

Architecture & Usage



```
Plane {
  vec T, R, S;
  id gpu_handle;
}
```

```
Model {
  vec T, R, S;
  string uri;
  buffer data;
}
```



Page
Tag Untagged Layer Default
Prefab Open Select Overrides

Transform

Position	X 0	Y 0	Z 0
Rotation	X 0	Y 0	Z 0
Scale	X 1	Y 1	Z 1

Page (Script)

Script: Page
Current Url: about:blank
Content Size: X 1920 Y 1080
Match CSS With Physical: ✓
Rendering Enabled: ✓

On Started (PageStartedEventArgs)
List is Empty

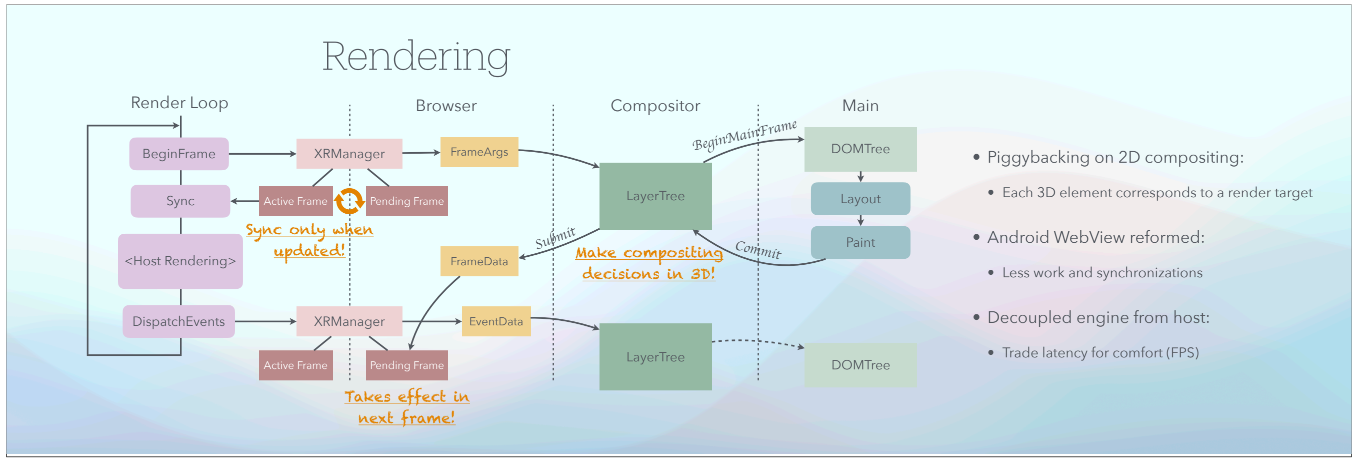
On Error (PageErrorEventArgs)
List is Empty

On Message (PageMessageEventArgs)
List is Empty

Each scene can contain multiple Web pages, with any one of them running WebXR immersive session.

Description data of planes and models are sent to the lower level engine, and with WebXR frame buffer they are composited.

Screenshot shows usage of our package in Unity: a prefab (WebView) that can be configured in ‘Unity style’.



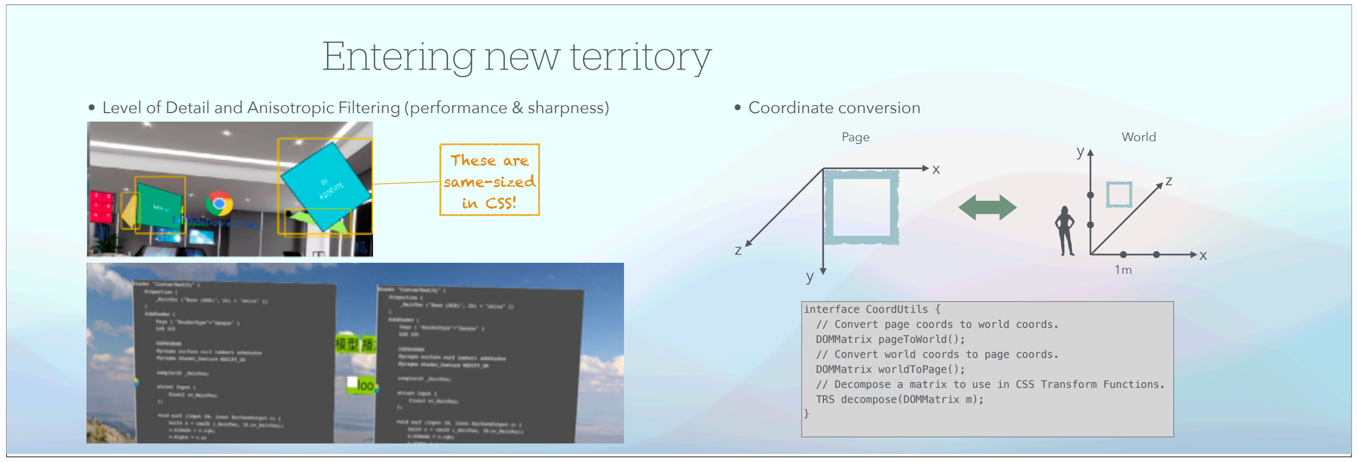
Rendering

- Piggybacking on 2D compositing:
  - Each 3D element corresponds to a render target
- Android WebView reformed:
  - Less work and synchronizations
- Decoupled engine from host:
  - Trade latency for comfort (FPS)

We modified the compositing step so that each plane or model is an individual render target, keeping its own transform in 3D.

Main render loop is made up of a couple of callbacks: (sending frame information to web engine) -> (receiving render targets from web engine) -> (render those targets along with other stuff of the host app)

XR rendering needs to run in high frame rate and must not block on Web rendering. So Web updates are async: an update can last for more than one frame, and takes effect by swapping the active frame and pending frame.


These are same-sized in CSS!
Page
x
y
z
World
y
z
x
1m

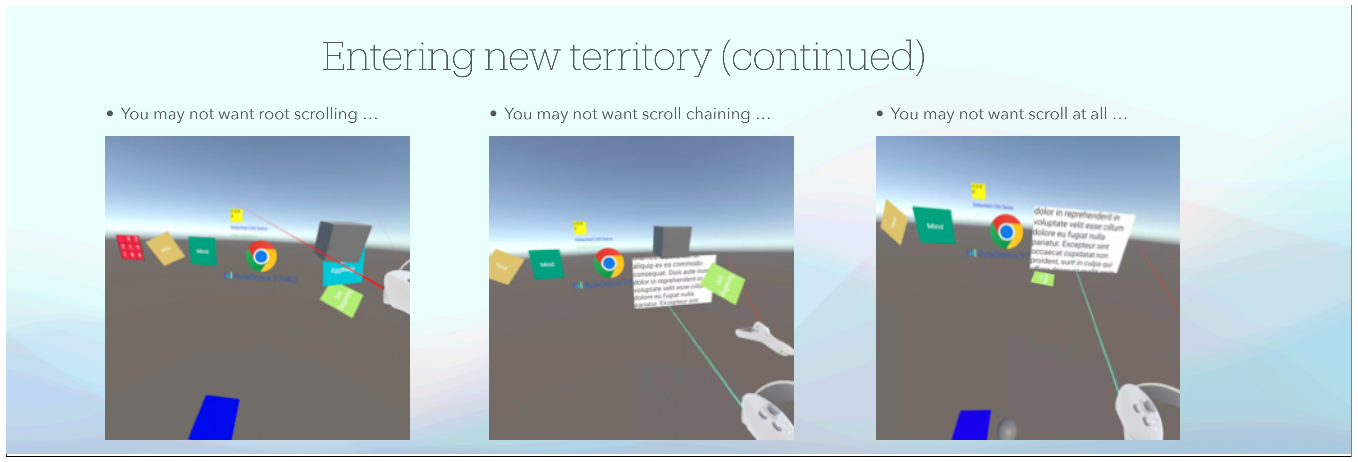
Entering new territory

- Level of Detail and Anisotropic Filtering (performance & sharpness)
- Coordinate conversion

```
interface CoordUtils {
  // Convert page coords to world coords.
  DOMMatrix pageToWorld();
  // Convert world coords to page coords.
  DOMMatrix worldToPage();
  // Decompose a matrix to use in CSS Transform Functions.
  TRS decompose(DOMMatrix m);
}
```

LOD and more 3D techniques are used to improve render perf and quality.

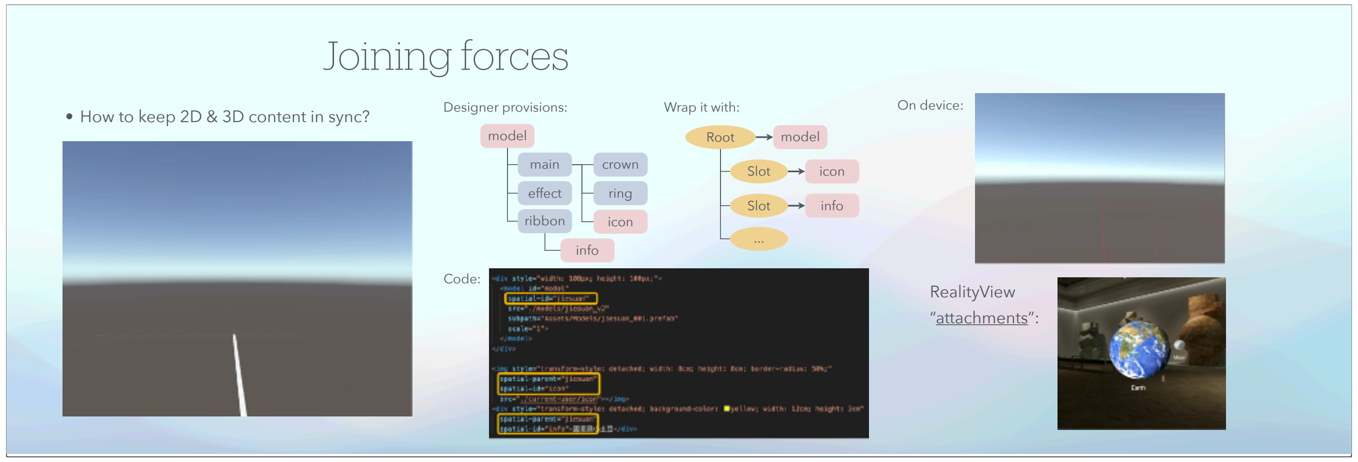
Util functions are provided to convert coords.



Entering new territory (continued)

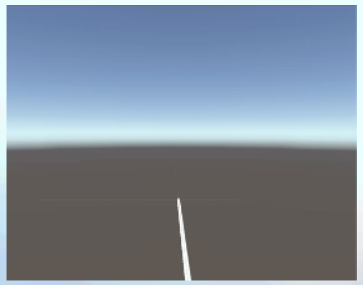
- You may not want root scrolling …
- You may not want scroll chaining …
- You may not want scroll at all …

Scrolling is hard in 3D. Design UX carefully.

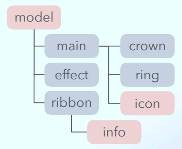


Joining forces

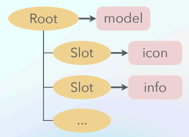
- How to keep 2D & 3D content in sync?



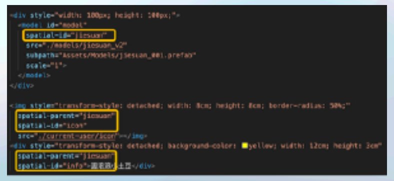
Designer provisions:



Wrap it with:

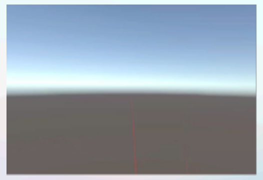


Code:

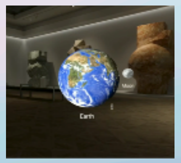


```
<div style="width: 100px; height: 100px;">
  <model id="model"
    spatial-id="jiesuan"
    src="./models/jiesuan_v2"
    subpath="Assets/Models/jiesuan_001.prefab"
    scale="1">
  </model>
</div>

<img style="transform-style: detached; width: 8cm; height: 8cm; border-radius: 50%;"
  spatial-parent="jiesuan"
  spatial-id="icon"
  src="./current-user/icon"></img>
<div style="transform-style: detached; background-color: yellow; width: 12cm; height: 3cm"
  spatial-parent="jiesuan"
  spatial-id="info">[illegible]</div>
```

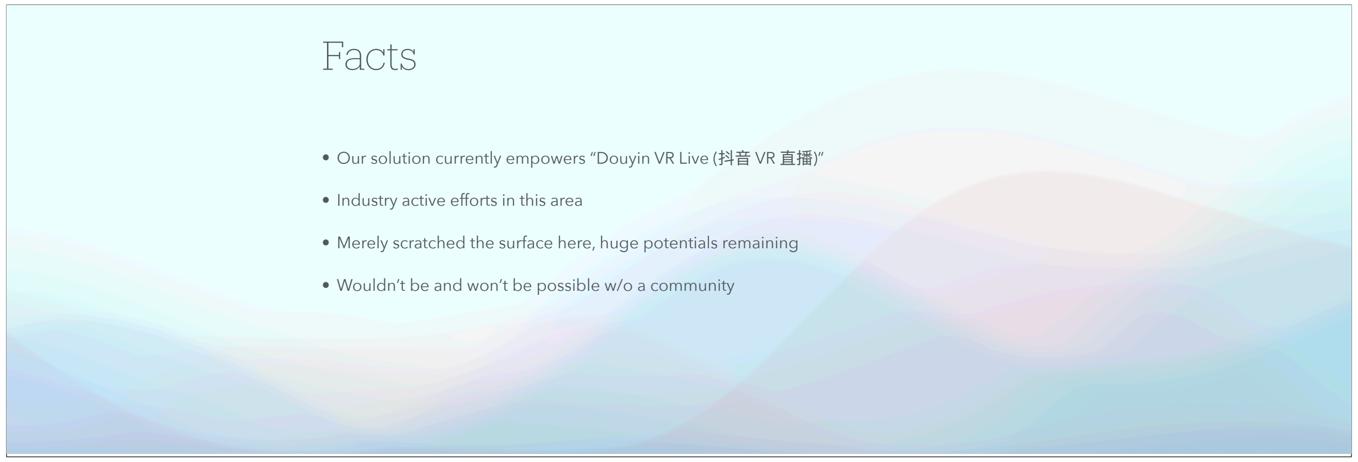
On device:



RealityView "attachments":



Sync planes with model animation: make slots on model and associate planes with slots at run time with declarative syntax.



Facts

- Our solution currently empowers “Douyin VR Live (抖音 VR 直播)”
- Industry active efforts in this area
- Merely scratched the surface here, huge potentials remaining
- Wouldn’t be and won’t be possible w/o a community

We believe the progressive path has huge potentials. And it is friendly in terms of accessibility, discoverability, and interoperability with existing Web.

Thank you!