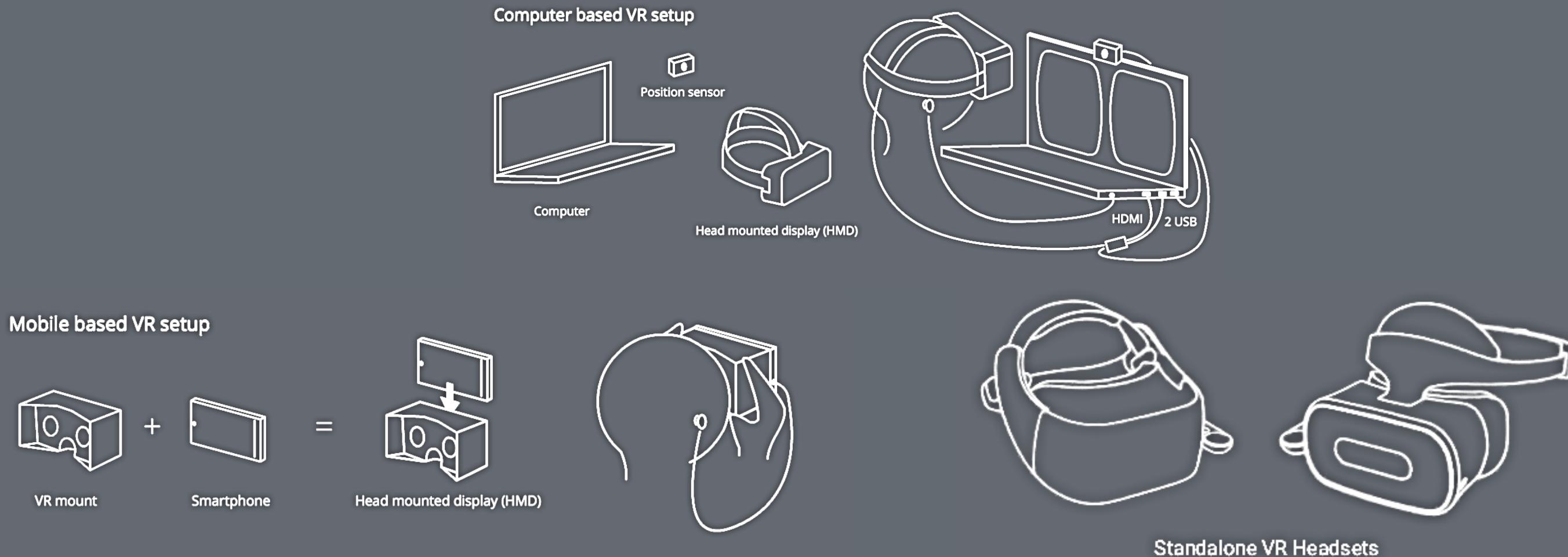


# Introduction to VR on the Web

# VR Setup\*



\* Illustrations from Mozilla

# VR Experiences

New considerations for VR app development

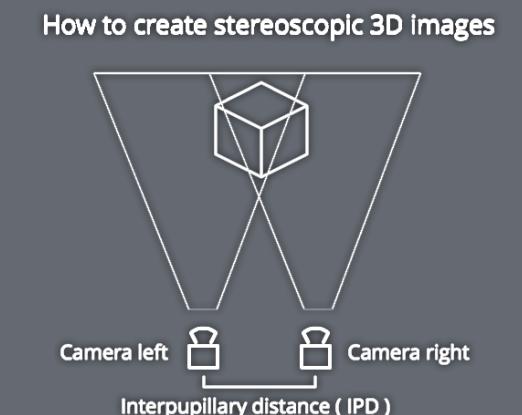
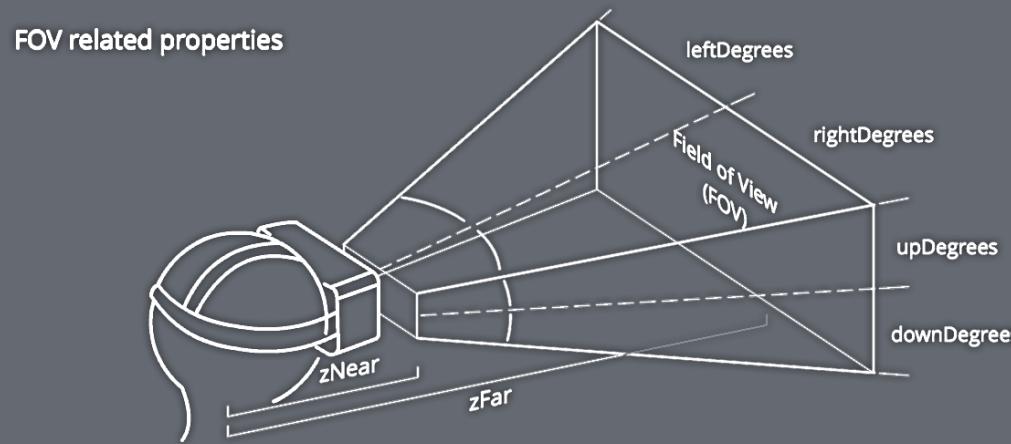
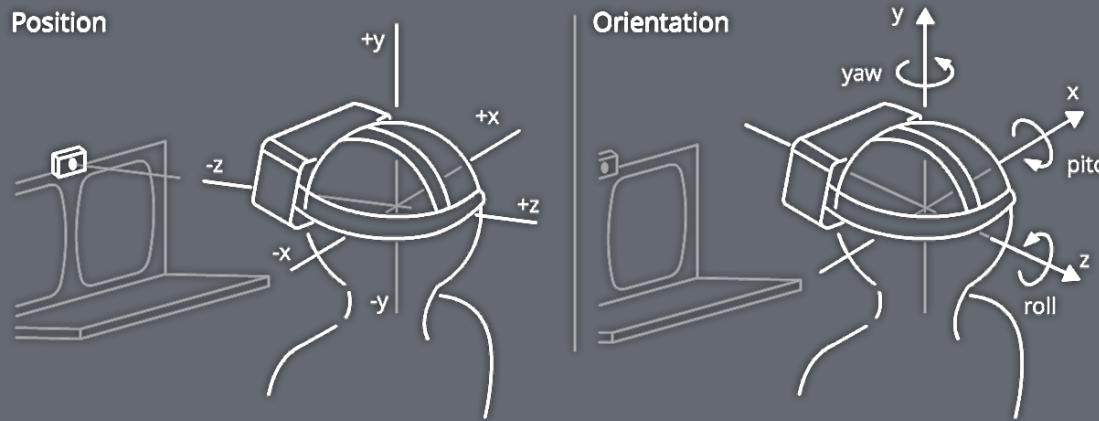
## **Focus on User Experience**

- Stereoscopic vision & interpupillary distance (IPD)
- Head tracking & degrees of freedom (DoF)
- Cone of focus & field of view (FOV)
- 3D positional audio
- Latency & frame rate (frames per second / FPS)

## **Focus on User Comfort**

- Eye strain
- Motion sickness

# Degrees of Freedom, Stereoscopic Vision & Interpupillary Distance\*



\* Illustrations from Mozilla

# Native VR Devices (PC & Console )



HTC Vive (room-scale VR), Oculus Rift, Playstation VR, Windows  
MR (Acer), Magic Leap, Fove, Razer OSVR

# Mobile VR Devices (Smartphone)



Samsung Gear VR  
Daydream  
Cardboard  
Hundreds of third-party headsets

# VR Programming

- Unreal Engine (C++)
- Unity (C#)
- Source Engine (C++)
- WebVR (WebXR Device API)

# VR Distribution

- SteamVR
- Google Play
- Oculus Store
- The Rift Arcade
- V "open alternative" to SteamVR

# Why WebVR?

- No app store ecosystem, distribution via internet
- Mobile & desktop automatically supported
- Uses current tools and libraries for JS
- Easy switch between VR & non-VR mode
- Interfacing with hardware through the browser



Progressive Enhancement by Arturo Paracuellos

# Virtual Reality on the Web

## A Retrospective 1/2

- 1994 VRML - first attempt to create an internet-based 3D language. (Mark Pesce presented the Labyrinth demo he developed with Tony Parisi and Peter Kennard.)
- VRML2 (1997) - added many features (animation) and later was succeeded by X3D  
Problem of VRML: plugin-based technology that only came preinstalled on IE

# Virtual Reality on the Web

## A Retrospective 2/2

- 2003 OpenGL ES - cross-language and multi-platform 3D graphics API.  
Hardware accelerated rendering of 3D objects.
- WebGL 1.0 introduced by Mozilla based on OpenGL ES uses DOM (canvas element)
- X3D ([web3d.org](http://web3d.org)) introduced **X3DOM**

# Most popular WebGL Engines

- Three.js by Ricardo Cabello in 2010
- Babylon.js
- p5\*js
- argon.js
- Scene.js
- Turbulenz

three.js



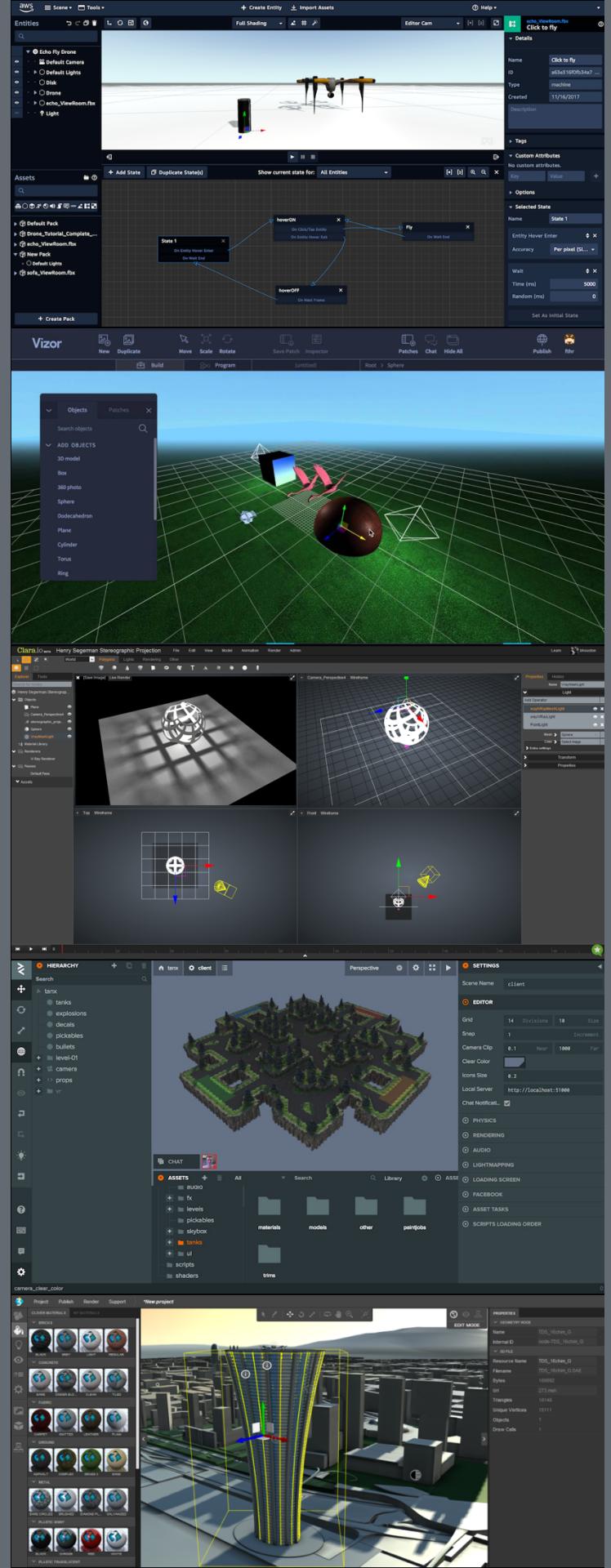
p5\*js

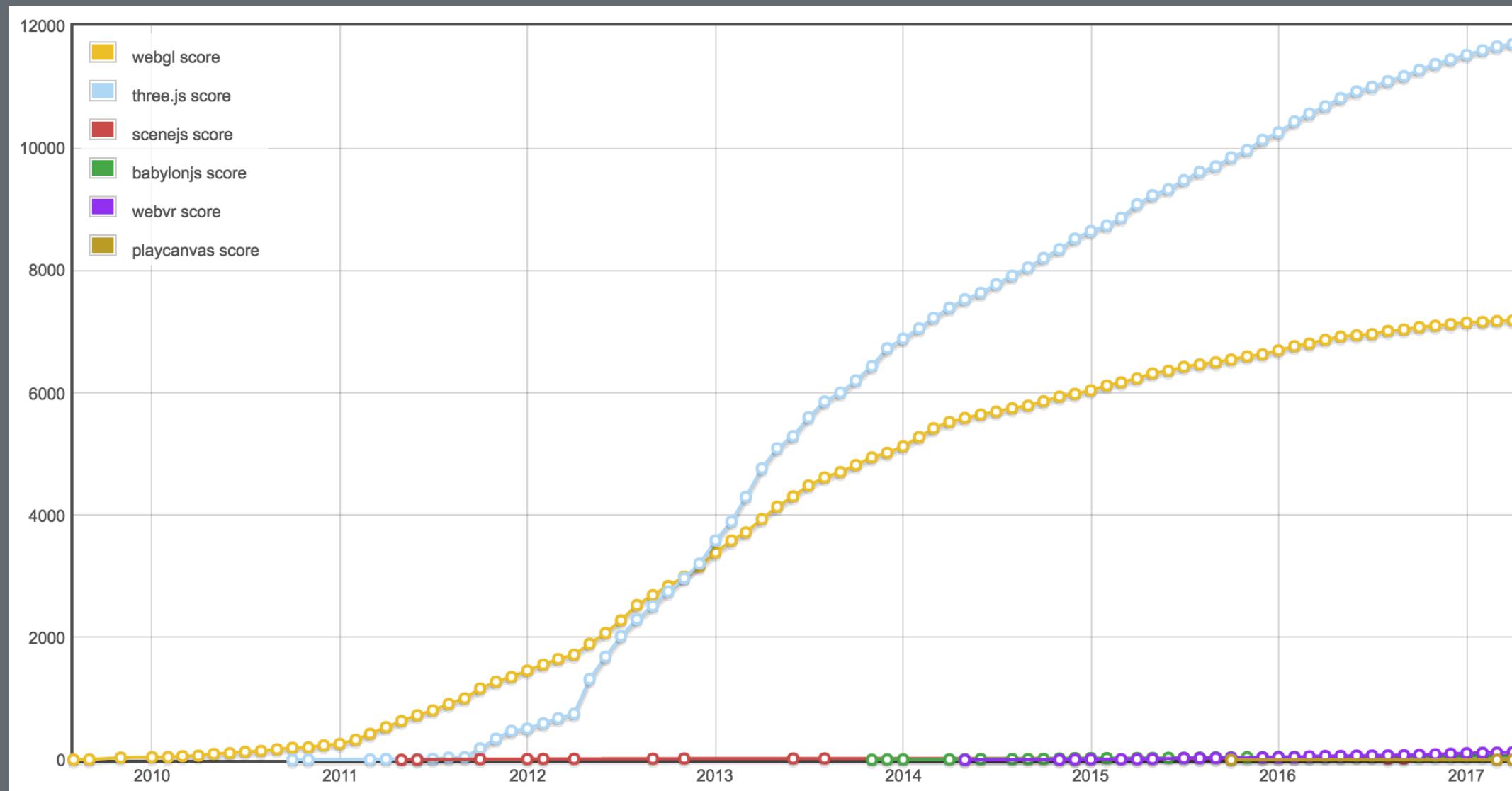
Ar



# Frameworks & Online WebVR Platforms

- A-Frame
- ReactVR
- Primrose
- Janus VR
- AWS Sumerian
- Vizor.io
- Clara.io
- Play Canvas
- Cl3ver





**Stackoverflow score growth over time by tag comparison**

WebGL Framework Comparison

# What is WebVR?

WebVR is a *experimental* Javascript API that provides interfaces to VR hardware to allow developers to build compelling, comfortable VR experiences on the web.

# WebVR

**WebVR v1.1 (Legacy) Editor's Draft, 12/12/17**

- deviceOrientation
- VRDisplay object handles almost everything
- available on multiple platforms

**WebXR Device API (WebVR 2.0) Editor's Draft, 01/11/18**

- not backwards compatible
- device support
- introduction of VRDevice & VRSession to replace VRDisplay
- read [Explainer doc!](#)

# Goals of WebVR

- support for accessing virtual reality (VR) and augmented reality (AR) devices, including sensors and head-mounted displays, on the Web.
- Detect available virtual reality devices.
- Query the device's capabilities.
- Poll the device's position and orientation.
- Display imagery on the device at the appropriate frame rate.

# Non-goals of WebVR

- Define how a virtual reality browser would work.
- Take full advantage of augmented reality devices.
- Build “The Metaverse.”

*Check out <https://webvr.rocks/> for support updates*

Demo Comparison A & B

```

15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
var isUserInteracting = false,
onMouseDownMouseX = 0, onMouseDownMouseY = 0,
lon = 0, onMouseDownLon = 0,
lat = 0, onMouseDownLat = 0,
phi = 0, theta = 0;

init();
animate();

function init() {

    var container, mesh;

    container = document.getElementById( 'container' );

    camera = new THREE.PerspectiveCamera( 75, window.innerWidth /
camera.target = new THREE.Vector3( 0, 0, 0 );

    scene = new THREE.Scene();

    var geometry = new THREE.SphereGeometry( 500, 60, 40 );
geometry.scale( - 1, 1, 1 );

    var material = new THREE.MeshBasicMaterial( {
        map: new THREE.TextureLoader().load( 'assets/sky.jpg' )
    } );

    mesh = new THREE.Mesh( geometry, material );

    scene.add( mesh );

    renderer = new THREE.WebGLRenderer();
}

function onDocumentMouseWheel( event ) {

    camera.fov += event.deltaY * 0.05;
    camera.updateProjectionMatrix();

}

function animate() {

    requestAnimationFrame( animate );
    update();

}

function update() {

    if ( isUserInteracting === false ) {

        lon += 0.05;

    }

    lat = Math.max( - 85, Math.min( 85, lat ) );
    phi = THREE.Math.degToRad( 90 - lat );
    theta = THREE.Math.degToRad( lon );

    camera.target.x = 500 * Math.sin( phi ) * Math.cos( theta );
    camera.target.y = 500 * Math.cos( phi );
    camera.target.z = 500 * Math.sin( phi ) * Math.sin( theta );
}

```



# Three.js and WebVR comparison with A-Frame

```

1 <!DOCTYPE html>
2 <html>
3     <head>
4         <title>A-Frame - Photosphere demo</title>
5         <meta name="description" content="A-Frame - Photosphere demo">
6         <meta charset="utf-8">
7         <meta name="viewport" content="width=device-width, user-scalable=no, minimum-scale=1.0, maximum-scale=1.0">
8         <script src="https://aframe.io/releases/0.5.0/aframe.min.js"></script>
9     </head>
10    <body>
11        <a-scene>
12            <a-sky src="assets/sky.jpg">
13                <a-animation attribute="rotation"
14                    dur="90000"
15                    fill="forwards"
16                    to="0 360 0"
17                    repeat="indefinite"
18                    easing="linear"></a-animation>
19            </a-sky>
20        </a-scene>
21    </body>
22 </html>

```

# A-Frame

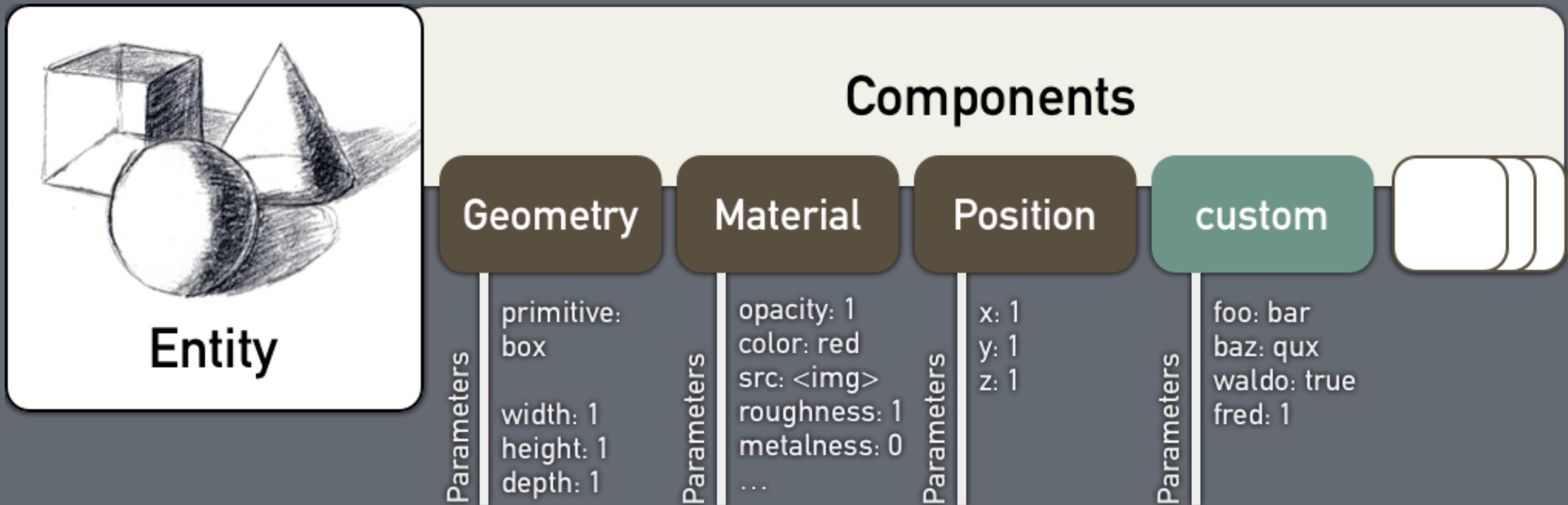
- a 3DML (3D markup language) like X3Dom and ReactVR
- DOM-based Entity-Component System => declarative & extensible (similar to Unity/UE4/PlayCanvas)

## The A-Frame Stack:



# A-Frame

## Entity-Component System



# A-Frame

- Entities are HTML elements (i.e., <a-entity>)
- Components are HTML attributes, set on the entity

```
<body>
  <a-scene>
    <a-entity geometry="primitive:box" material="color:#c00">
    </a-entity>
  </a-scene>
</body>
```

# Entity

- general purpose objects (e.g. create a player, ball, or field)
- they inherently have a position, rotation and scale in a scene

```
<a-entity
  geometry="primitive:box;
    width:0.2;
    height:0.3;
    depth:0.5;"
  material="color:#c00"
  position="0 0 -1"
  rotation="0 30 30"
  material="color:#c00">
</a-entity>
```

# Primitives (concise, semantic building blocks)

```
<a-entity geometry="primitive:box;  
          width:0.2;  
          height:0.3;  
          depth:0.5;"  
        material="color:#c00">  
</a-entity>  
  
<a-box width="0.2"  
      height="0.3"  
      depth="0.5"  
      material="color:#c00">  
</a-box>
```

# Mixins

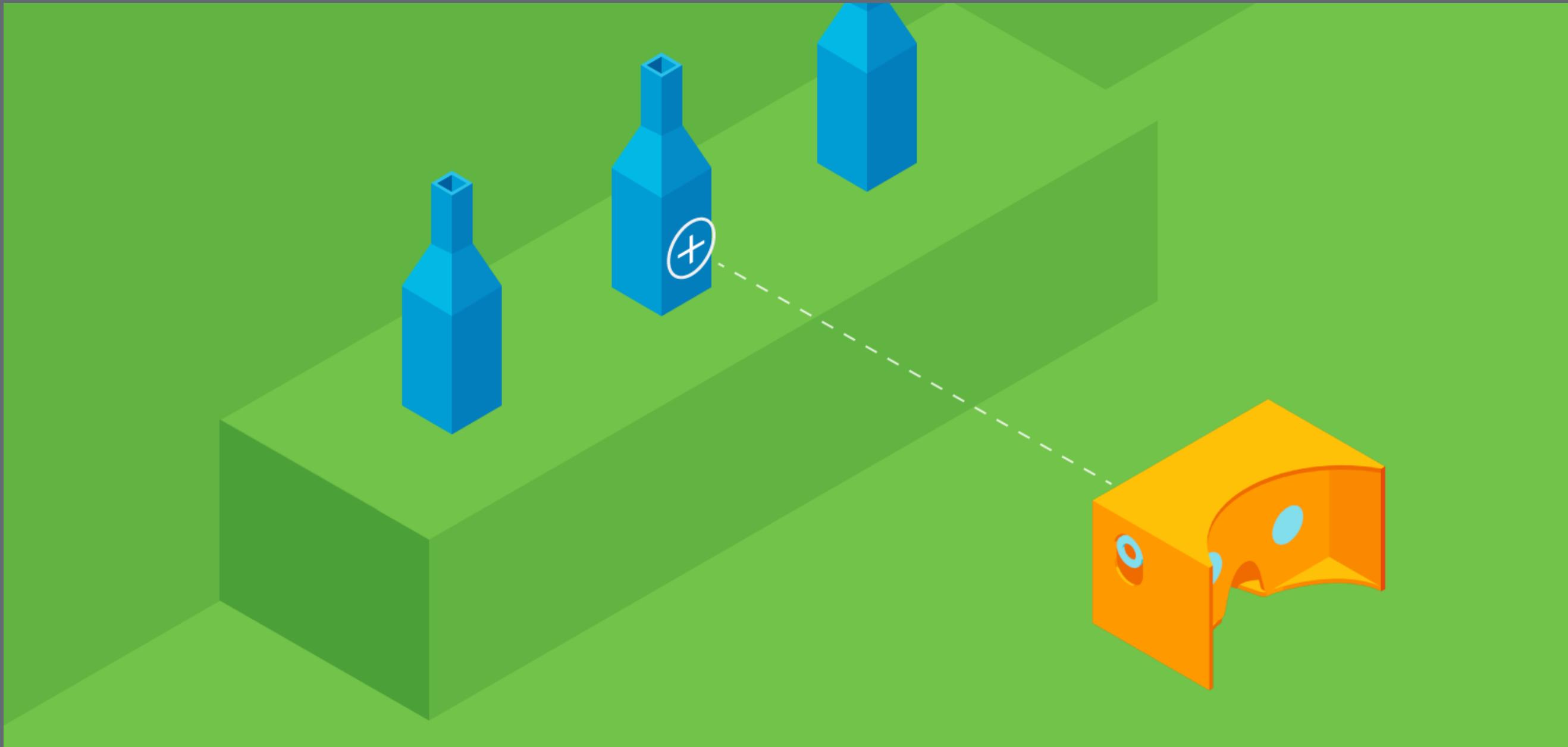
```
<a-mixin id="box"  
         geometry="primitive:box;  
                  width:0.2;  
                  height:0.3;  
                  depth:0.5;"  
         material="color:#c00">  
</a-mixin>
```

```
<a-entity mixin="box">  
</a-entity>
```

# Custom Components

```
AFRAME.registerComponent('foo', {  
  schema: {  
    bar: {type: 'number'},  
    baz: {type: 'string'}  
  },  
  init: function () {  
    // Do something when component is plugged in.  
  },  
  update: function () {  
    // Do something when component's data is updated.  
  }  
});
```

# Gaze-Based Control



# Gaze-Based Cursor Interactions

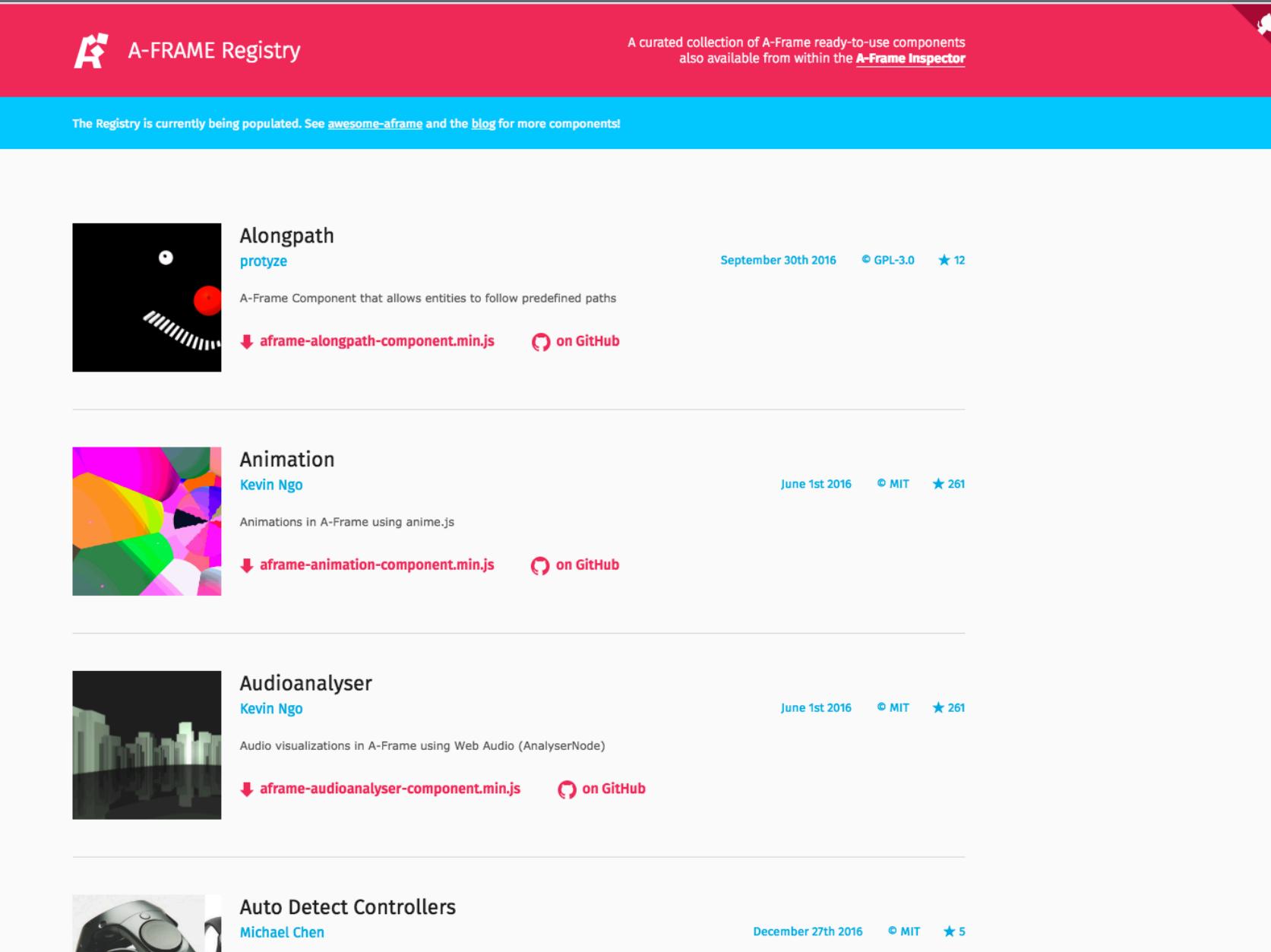
```
AFRAME.registerComponent('clickable', {
  init: function () {
    var el = this.el;
    el.addEventListener('mouseenter', function () {
      el.setAttribute('color', '#f00');
    });
    el.addEventListener('mouseleave', function () {
      el.setAttribute('color', '#fff');
    });
    el.addEventListener('click', function () {
      el.setAttribute('material', 'src', 'assets/football.png');
    });
  }
});

<a-sphere clickable src="assets/basketball.png" radius="0.5" color="#fff"></a-sphere>
<a-camera><a-cursor></a-cursor></a-camera>
```

# A-Frame Component Registry

The Registry is currently being populated. See [awesome-aframe](#) and the [blog](#) for more components!

---



**Alongpath**  
protoype  
A-Frame Component that allows entities to follow predefined paths  
[aframe-alongpath-component.min.js](#) [on GitHub](#)

**Animation**  
Kevin Ngo  
Animations in A-Frame using anime.js  
[aframe-animation-component.min.js](#) [on GitHub](#)

**Audioanalyser**  
Kevin Ngo  
Audio visualizations in A-Frame using Web Audio (AnalyserNode)  
[aframe-audioanalyser-component.min.js](#) [on GitHub](#)

**Auto Detect Controllers**  
Michael Chen  
December 27th 2016 [© MIT](#) ★ 5

[DOCS](#)[EXAMPLES](#)[BLOG](#)[COMMUNITY](#)[GITHUB](#)[SLACK](#)

VERSION 0.5.0

[INTRODUCTION](#)[Introduction](#)[Getting Started](#)[Device & Platform Support](#)[Best Practices](#)[FAQ](#)[GUIDES](#)[Building a Basic Scene](#)

A-Frame is a web framework for building virtual reality experiences. It was started by [Mozilla VR](#) to make [WebVR](#) content creation easier, faster, and more accessible.

A-Frame lets you build scenes with just [HTML](#) while having unlimited access to JavaScript, [three.js](#), and all existing Web APIs. A-Frame uses an **entity-component-system** pattern that promotes composition and extensibility. It is free and open source with a welcoming community and a thriving [ecosystem of tools and components](#).

<https://aframe.io/docs/0.7.0/>

HTML is one of the easiest languages to understand, and many of us are already familiar with it. There are no build steps or boilerplate required nor anything to install; all we need is an HTML file:

```
<html>
  <head>
    <script src="https://aframe.io/releases/0.5.0/aframe.min.js"></script>
  </head>
  <body>
    <a-scene>
      <a-box color="#6173F4" opacity="0.8" depth="2"></a-box>
      <a-sphere radius="2" src="texture.png" position="1 1 0"></a-sphere>
      <a-sky color="#ECECEC"></a-sky>
```

HTML

[Back to Scene](#)

The screenshot shows the A-Frame Inspector interface. At the top, there are various icons for scene management. On the right side, the properties panel is open for a component named <a-sky>. The properties listed are:

ID	position	rotation	scale	visible	mixins
	0.000 0.000 0.000	0.000 0.000 0.000	-1.000 1.000 1.000	<input checked="" type="checkbox"/>	box <input type="button" value="+"/>

Below these, there are buttons for "Add component..." and "Copy Attributes". Under the "GEOMETRY" section, the following attributes are shown:

buffer	mergeTo	primitive	skipCache	radius	phiLength	phiStart	thetaLength	thetaStart
<input checked="" type="checkbox"/>		sphere <input type="button" value="x"/>	<input type="checkbox"/>	5000.000	360.000	0.000	180.000	0.000

A large, semi-transparent text overlay in the center of the screen reads "A-Frame Inspector [control]+[option]+[i]".



Create responsibly!

# Resources

- \* [A-Frame Slack](#)
- \* [A-Frame](#)
- \* [A-Frame Component Registry](#)
- \* [vr.mozilla.org](#)
- \* [VR Gets Real with WebVR](#)
- \* [WebVR API](#)
- \* [The UX of VR](#)

## Get In Touch

@rolanddubois

[rolanddubois.com](http://rolanddubois.com)

# *It's code time!* Here's what we're building:

- [github.com/roland-dubois/aframe-meetup-nyc](https://github.com/roland-dubois/aframe-meetup-nyc)
- XAMPP, MAMP or \$ python -m SimpleHTTPServer
- JSBin or Glitch

