PSCR 2021

THE DIGITAL EXPERIENCE

#PSCR2021 • PSCR.GOV

NIST

PSCR

AN INTRODUCTION TO WEBXR

Paul Merritt - Lead AR Developer

NIST

#PSCR2021

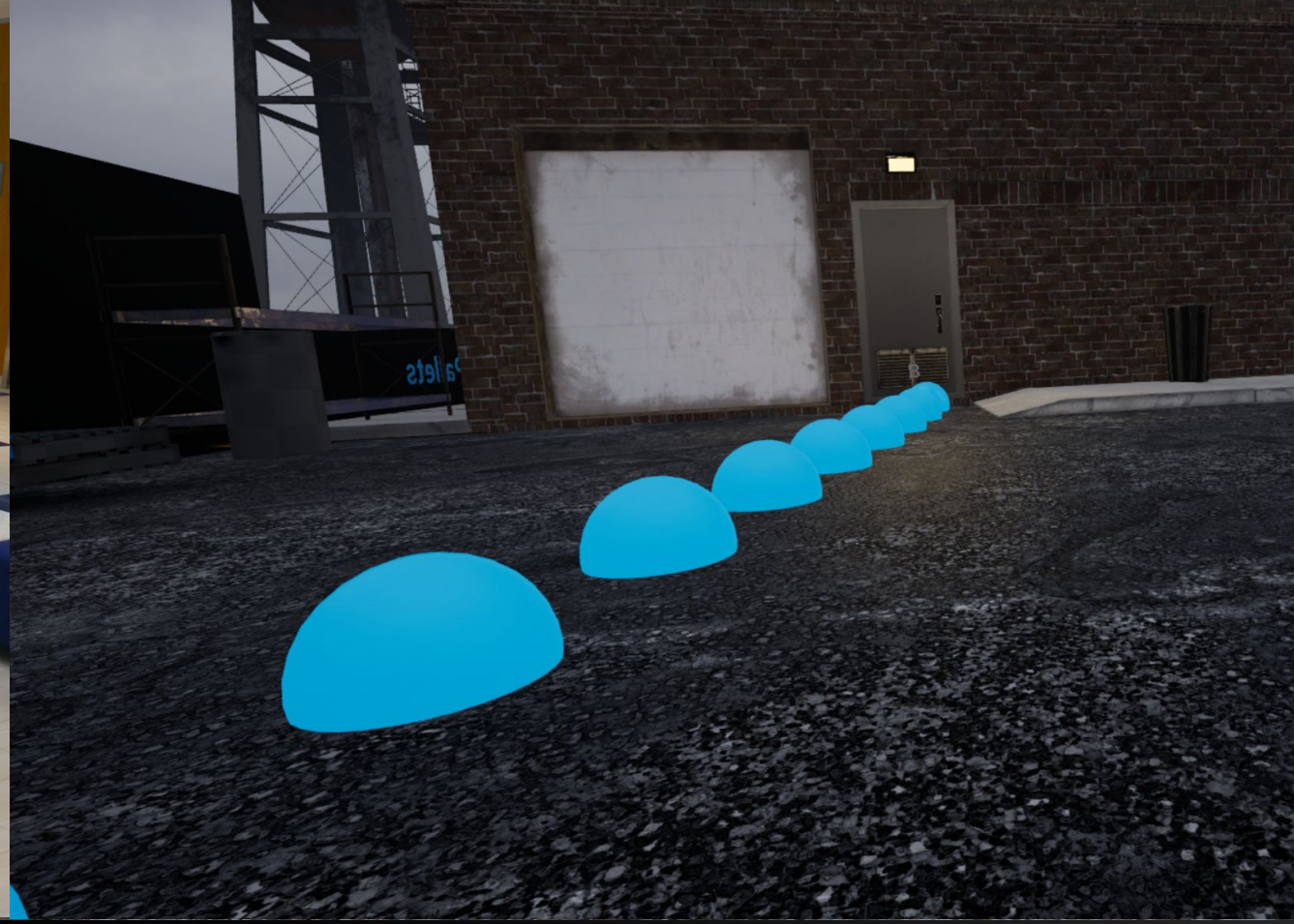
PSCR

DISCLAIMER

Certain commercial entities, equipment, or materials may be identified in this document in order to describe an experimental procedure or concept adequately.

Such identification is not intended to imply recommendation or endorsement by the National Institute of Standards and Technology, nor is it intended to imply that the entities, materials, or equipment are necessarily the best available for the purpose.

*** Please note, unless mentioned in reference to a NIST Publication, all information and data presented is preliminary/in-progress and subject to change**



AR **vs** **VR**

Virtual objects superimposed on the real world

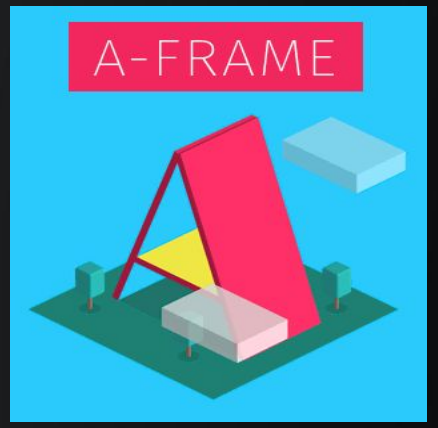
A completely virtual space

WHAT IS WEBXR?

- A platform combining AR and VR (mi**X**ed **R**eality)
  - Easy to use
    - Similar to HTML
    - Convenient
    - Various frameworks

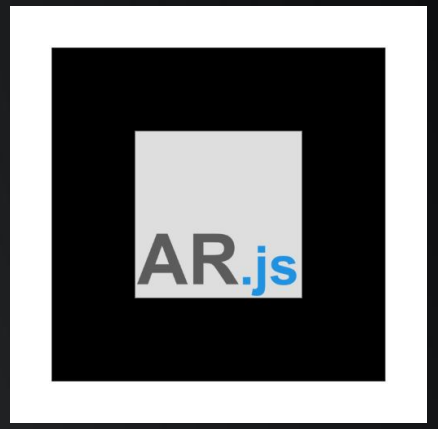
Images taken from blog.mozilla.org and learn.framevr.io

A-FRAME AND AR.JS



A-Frame

- Entity component structure
- Document Object Model (DOM) based, but performant



AR.js

- Lightweight
- Adds location/marker-based functionality

WALKING THROUGH YOUR FIRST WEBXR APP

NIST

#PSCR2021

PSCR

GETTING STARTED WITH A-FRAME

This slide is a video demonstrating how to create a basic WebXR scene using A-Frame. If you cannot view this video, please view my recorded presentation at pscr.gov

GETTING STARTED WITH A-FRAME

```html
<html>
  <head>
    <script src="https://aframe.io/releases/1.2.0/aframe.min.js">

    </script>
  </head>
  <body>
    <a-scene>
    </a-scene>
  </body>
</html>
```

Just a few lines

Initialize by referencing the script in the header

Populate the scene in the body

Entity objects!

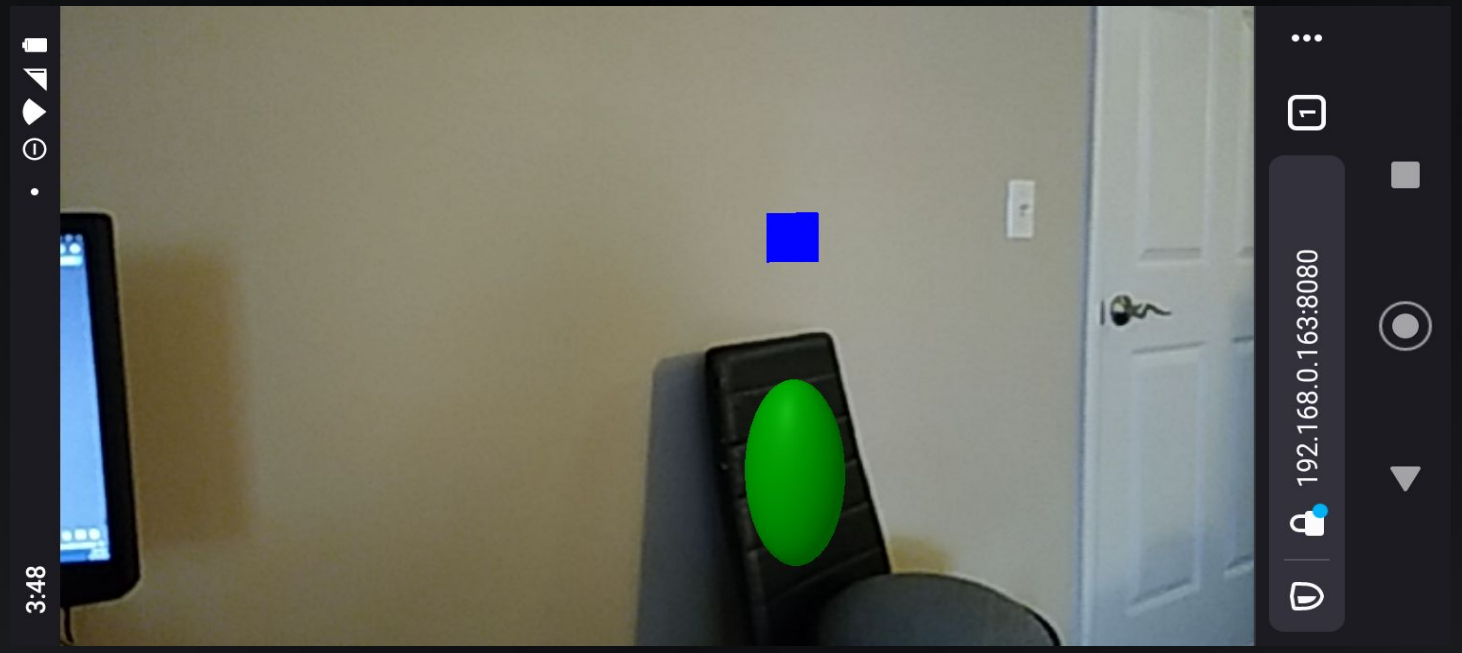
```html
<body>

  <a-scene>

      <a-sphere position="0 1 -3" scale=".1 .2 .1" material="color: green;"></a-sphere>

      <a-box position="0 1.5 -3" scale=".1 .1 .1" material="color: blue;"></a-box>

  </a-scene>

</body>
```

BASIC A-FRAME SCENE

```
<body style='margin : 0px; overflow: hidden;'>
  {{!-- a-frame scene for the objects we place --}}
  <a-scene>

        <a-sphere position="0 1 -3" scale=".1 .2 .1" material="color: green;"></a-sphere>
        <a-box position="0 1.5 -3" scale =".1 .1 .1" material="color: blue;"></a-box>

  </a-scene>
</body>
```

BASIC A-FRAME SCENE

```
<body style='margin : 0px; overflow: hidden;'>
  {{!-- a-frame scene for the objects we place --}}
  <a-scene>

        <a-sphere position="0 1 -3" scale=".1 .2 .1" material="color: green;"></a-sphere>
        <a-box position="0 1.5 -3" scale =".1 .1 .1" material="color: blue;"></a-box>

  </a-scene>
</body>
```



ADDING A CURSOR FOR INTERACTIVITY

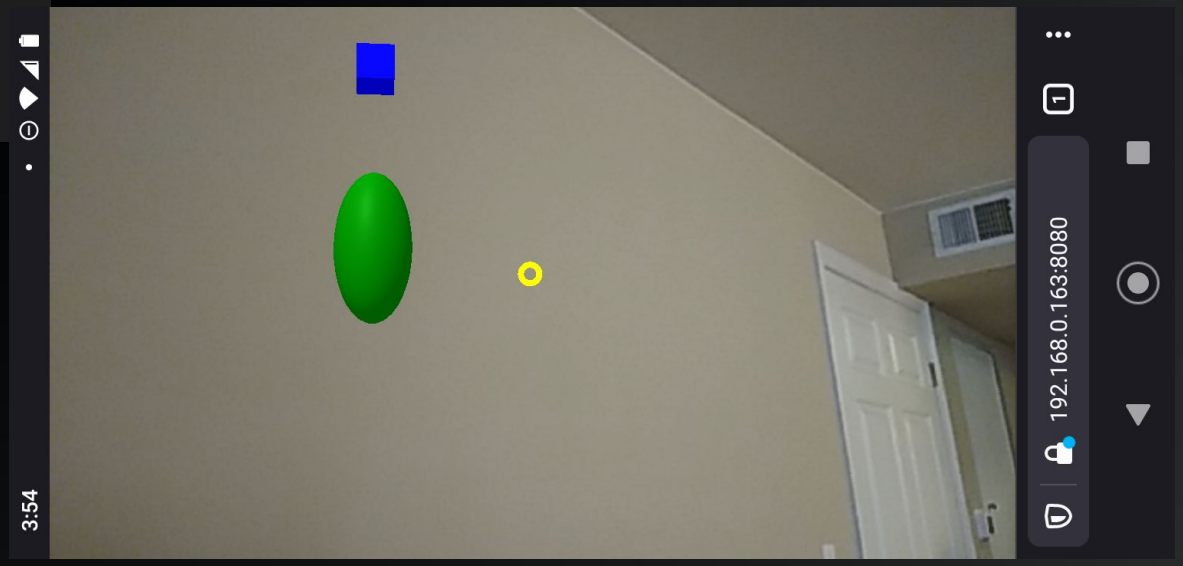
This slide is a video demonstrating how to add a cursor to your scene. If you cannot view this video, please view my recorded presentation at pscr.gov

ADDING A CURSOR FOR INTERACTIVITY

```
<body style='margin : 0px; overflow: hidden;'>
  {{!-- a-frame scene for the objects we place --}}
  <a-scene>

    <a-camera camera="" position="" rotation="" look-controls="" wasd-controls="">
        <a-entity cursor="fuse: true; fuseTimeout: 500"
                  position="0 0 -1"
                  geometry="primitive: ring; radiusInner: 0.005; radiusOuter: 0.01"
                  material="color: yellow; shader: flat">
      </a-entity>
    </a-camera>



        <a-sphere position="0 1 -3" scale=".1 .2 .1" material="color: green;"></a-sphere>
        <a-box position="0 1.5 -3" scale =".1 .1 .1" material="color: blue;"></a-box>

  </a-scene>
</body>
```

ADDING A CURSOR FOR INTERACTIVITY

```
<body style='margin : 0px; overflow: hidden;'>
  {{!-- a-frame scene for the objects we place --}}
  <a-scene>

    <a-camera camera="" position="" rotation="" look-controls="" wasd-controls="">
        <a-entity cursor="fuse: true; fuseTimeout: 500"
                  position="0 0 -1"
                  geometry="primitive: ring; radiusInner: 0.005; radiusOuter: 0.01"
                  material="color: yellow; shader: flat">
      </a-entity>
    </a-camera>



        <a-sphere position="0 1 -3" scale=".1 .2 .1" material="color: green;"></a-sphere>
        <a-box position="0 1.5 -3" scale =".1 .1 .1" material="color: blue;"></a-box>

  </a-scene>
</body>
```



ADDING INTERACTIVITY

This slide is a video demonstrating how to add event handling for that cursor. If you cannot view this video, please view my recorded presentation at pscr.gov

ADDING INTERACTIVITY

```
    <script type="text/javascript">

    AFRAME.registerComponent('cursor-listener', {
  init: function () {
    this.el.addEventListener('click', function (evt) {
      if (this.getAttribute('clicked') == "0"){

        this.setAttribute('material', 'color: green');
        this.setAttribute('clicked', "1");

      }
      else if (this.getAttribute('clicked') == "1"){

        this.setAttribute('material', 'color: blue');
        this.setAttribute('clicked', "0");

      }
    });
  }
});
    </script>

  </head>
  <body style='margin : 0px; overflow: hidden;'>
    {{!-- a-frame scene for the objects we place --}}
    <a-scene
    vr-mode-ui="enabled: false"
      embedded
      arjs="sourceType: webcam; debugUIEnabled: false; detectionMode: mono_and_matrix; matrixCodeType: 3x3;"
    >

      <a-marker-camera camera="" position="" rotation="" look-controls="" wasd-controls="">
          <a-entity cursor="fuse: true; fuseTimeout: 500"
                    position="0 0 -1"
                    geometry="primitive: ring; radiusInner: 0.005; radiusOuter: 0.01"
                    material="color: yellow; shader: flat">
        </a-entity>
      </a-marker-camera>

            <a-sphere position="0 1 -3" scale=".1 .2 .1" cursor-listener clicked="0" material="color: red;"></a-sphere>

            <a-box position="0 -.5 -3" scale =".1 .1 .1" cursor-listener clicked="0" material="color: red;"></a-box>

    </a-scene>
  </body>
</html>
```

ADDING INTERACTIVITY

This slide is a video demonstrating what that interactivity looks like. If you cannot view this video, please view my recorded presentation at pscr.gov

INJECTING AR.JS

```
<body>

  <a-scene
    vr-mode-ui="enabled: false"
    embedded
    arjs="sourceType: webcam; debugUIEnabled: false;"
  >
    <a-text
      value="This content will always face you."
      look-at="[gps-camera]"
      scale="120 120 120"
      gps-entity-place="latitude: <add-your-latitude>; longitude: <add-your-longitude>;"
    ></a-text>
    <a-camera gps-camera rotation-reader> </a-camera>
  </a-scene>

</body>
```

An extension of A-Frame

Introduces a few entities and components

Adds fundamental AR functionalities

Location-Based and Image-Tracking

```
<body>

  <a-scene embedded arjs>
      <a-marker preset="hiro">
          <a-entity
          position="0 -1 0"
          scale="0.05 0.05 0.05"
          gltf-model="https://arjs-cors-proxy.herokuapp.com/https://
          AR.js/master/aframe/examples/image-tracking/nft/trex/scene
          ></a-entity>
      </a-marker>
      <a-entity camera></a-entity>
      </a-scene>

</body>
```

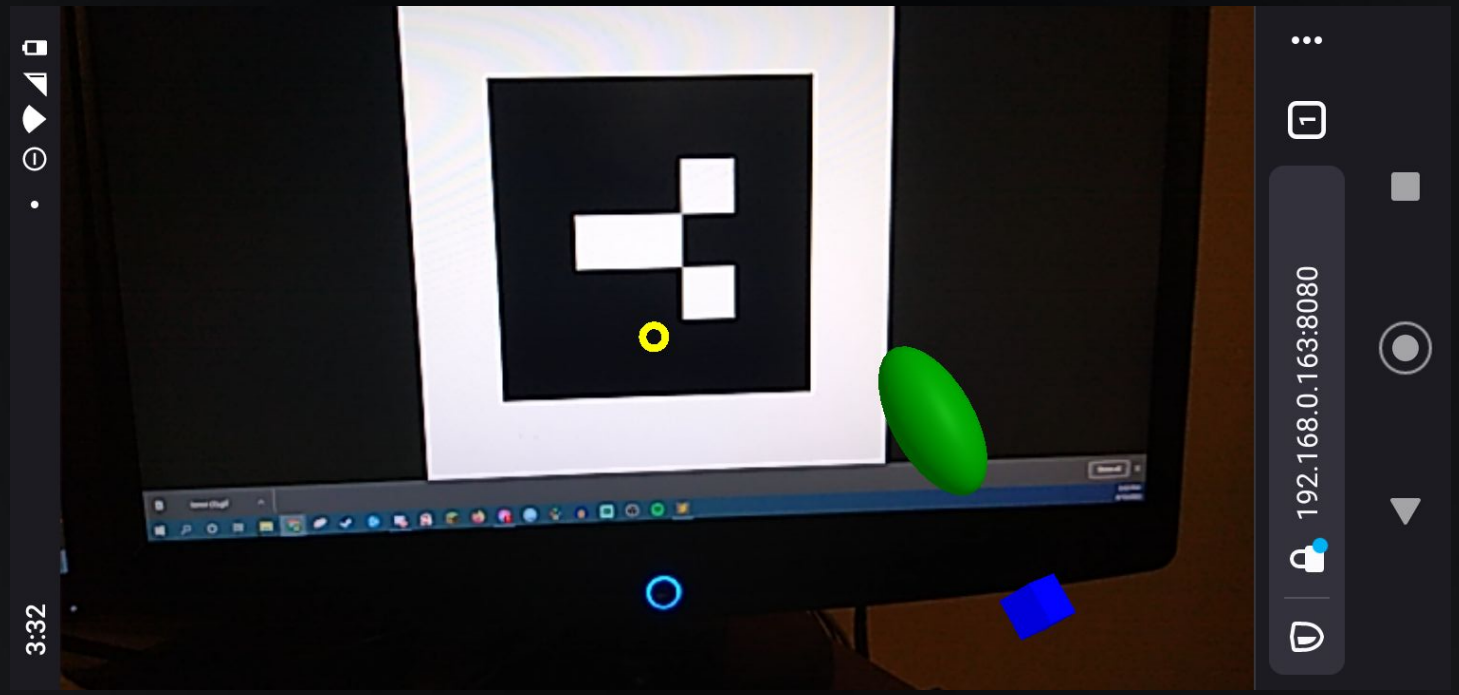
ADDING A MARKER WITH AR.JS

This slide is a video showcasing what that interactivity looks like. If you cannot view this video, please view my recorded presentation at pscr.gov

ADDING A MARKER WITH AR.JS

```
<body style='margin : 0px; overflow: hidden;'>
  {{!-- a-frame scene for the objects we place --}}
  <a-scene
  vr-mode-ui="enabled: false"
    embedded
    arjs="sourceType: webcam; debugUIEnabled: false; detectionMode: mono_and_matrix; matrixCodeType: 3x3;"
  >

    <a-marker-camera camera="" position="" rotation="" look-controls="" wasd-controls="">
        <a-entity cursor="fuse: true; fuseTimeout: 500"
                  position="0 0 -1"
                  geometry="primitive: ring; radiusInner: 0.005; radiusOuter: 0.01"
                  material="color: yellow; shader: flat">
      </a-entity>
    </a-marker-camera>

    <a-marker type="barcode" value="35">

        <a-sphere position="0 1 -3" scale=".1 .2 .1" material="color: green;"></a-sphere>

        <a-box position="0 -.5 -3" scale =".1 .1 .1" material="color: blue;"></a-box>

    </a-marker>

  </a-scene>
</body>
```

ADDING A MARKER WITH AR.JS



WHAT DOES IT MEAN THAT IT IS ENTITY BASED?

DYNAMIC

DOCUMENT OBJECT MODEL (DOM) MANIPULATION

querySelector()

accessing any element by id/class

createElement()

adding an element to the a-scene at runtime

getAttribute()

accessing an attribute of that element

setAttribute()

changing an attribute at runtime

TEMPLATING

Templating friendly

```
{{#each sources_array}}
    <a-marker type="barcode" value={{this.marker}}>
      <a-box
      font="roboto"
      scale="3 3 .1"
      color={{this.color}}
      opacity=".8"
      look-at="[camera]"
      >
      {{!-- id is given based on the port to allow us to iterate and replace text value --}}
        <a-text id ={{this.port}}
            value="replace me please"
            look-at="[camera]"
            position="0 0 5"
            scale=".15 .15 .15"
            align="center"
        >
        </a-text>
      </a-box>
    </a-marker>
{{/each}}
```

AJAX

```
<script type="text/javascript">

//run my_function every second
setInterval("my_function();",1000);
function my_function(){
    //AJAX request for most recent variable data
    var xhttp = new XMLHttpRequest();
    xhttp.onreadystatechange = function() {
        if (this.readyState == 4 && this.status == 200) {
            var myArr = JSON.parse(this.responseText);
            myArr.forEach(source => {
              //update the text for each source with the most recent text
              var t = document.getElementById(''+source.port);
              t.setAttribute('value', ''+source.out);
            });
        }
    };
    xhttp.open("GET", "/sources", true);
    xhttp.send();
}
</script>
```

Easy to edit at runtime

UPDATING TEXT DATA WITH AJAX

This slide is a video teaching how to use AJAX to update the contents of your scene. If you cannot view this video, please view my recorded presentation at pscr.gov

UPDATING TEXT DATA WITH AJAX

```
function update_text(ele){
    var xhttp = new XMLHttpRequest();
    xhttp.onreadystatechange = function(){
        if (this.readyState == 4 && this.status==200){
            var myArr = JSON.parse(this.responseText);
            ele.setAttribute('text', 'value: ' + JSON.stringify(myArr[0]) + "color: black; align:center; wrapCount:100; width:
                auto; height: auto;");
        }
    }
    xhttp.open("GET", "/sources", true);
    xhttp.send();
}
AFRAME.registerComponent('cursor-listener', {
    init function() {
        this.el.addEventListener('click', function(evt){
            if (this.getAttribute('clicked') == "0"){
                update_text(this);
                this.setAttribute('material', 'color: gray');
                this.setAttribute('clicked', "1");
            }
            else {
                this.setAttribute('material', 'color: red');
                this.setAttribute('text', 'value: click me');
                this.setAttribute('clicked', "0");
            }
        });
    }
});
```

UPDATING TEXT DATA WITH AJAX

```
<body>
    <a-scene
    arjs="sourceType: webcam; debugUIEnabled: false; detectionMode: mono_and_matrix; matrixCodeType: 3x3;">

        <a-marker-camera>
            <a-entity cursor="fuse: true; fuseTimeout: 500"
            position="0 0 -1"
            geometry="primitive: ring; radiusInner: 0.005; radiusOuter: 0.01"
            material="color: yellow; shader: flat;">
            </a-entity>
        </a-marker-camera>

        <a-marker type="barcode" value="35">

            <a-entity cursor-listener clicked="0" position="0 0 -1"
            text="value: click me; color: black; align:center; wrapCount:100; width: auto; height: auto;"
            geometry="primitive: plane; width: auto; height: auto;"
            material="color: #DDDDDD; shader: flat;">
            </a-entity>

        </a-marker>
    </a-scene>
</body>
```


Sep 19 2020
FIRE SCENARIO
08:00:52
265
2
1
32
5
1

PULLING
THE FUTURE
FORWARD

WEBXR AND ITS FUTURE

Easy to put together

HTML/JavaScript, plenty of examples

Convenient for any device

First Responder teams with multiple devices can access and share information with ease

CloudXR, OpenXR

Complex WebXR platforms are here and coming soon

THANK YOU

#PSCR2021 • PSCR.GOV