PRO-C167

IMAGE TRACKING AR

What is our GOAL for this MODULE?

We learned about image tracking based on augmented reality and created a web-based image tracker AR and played a video using the image marker.

What did we ACHIEVE in the class TODAY?

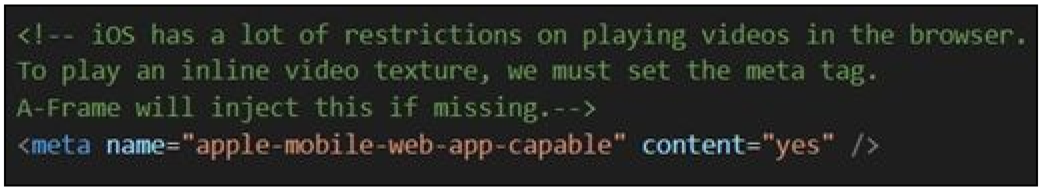
- Learned about image tracking augmented reality web apps.
- Learned to create a basic web-based AR app using Image trackers
- Learned to play video as AR scenes.

Which CONCEPTS/CODING BLOCKS did we cover today?

- aframe-ar-nft.js library
- <a-nft> tag ,<video>,<a-video> tags
- ngrok to run the application.

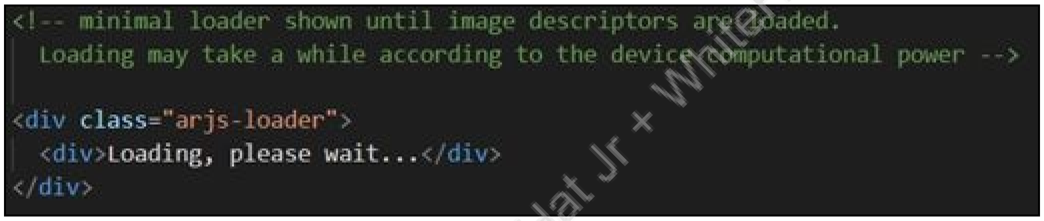
How did we DO the activities?

1. Set up the meta information in the <head> tag.

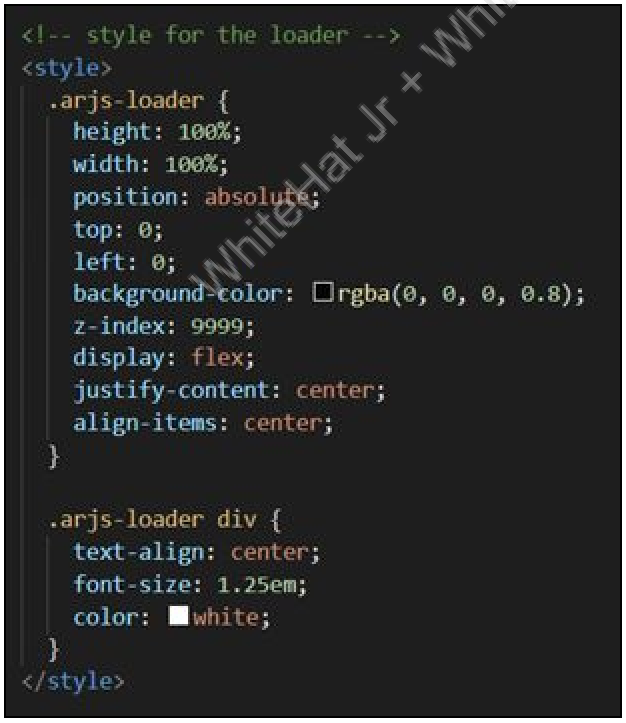


```
<!-- iOS has a lot of restrictions on playing videos in the browser.
To play an inline video texture, we must set the meta tag.
A-Frame will inject this if missing.-->
<meta name="apple-mobile-web-app-capable" content="yes" />
```

2. Add <div> in the <body> to show the loading descriptor till the time video content is loaded.



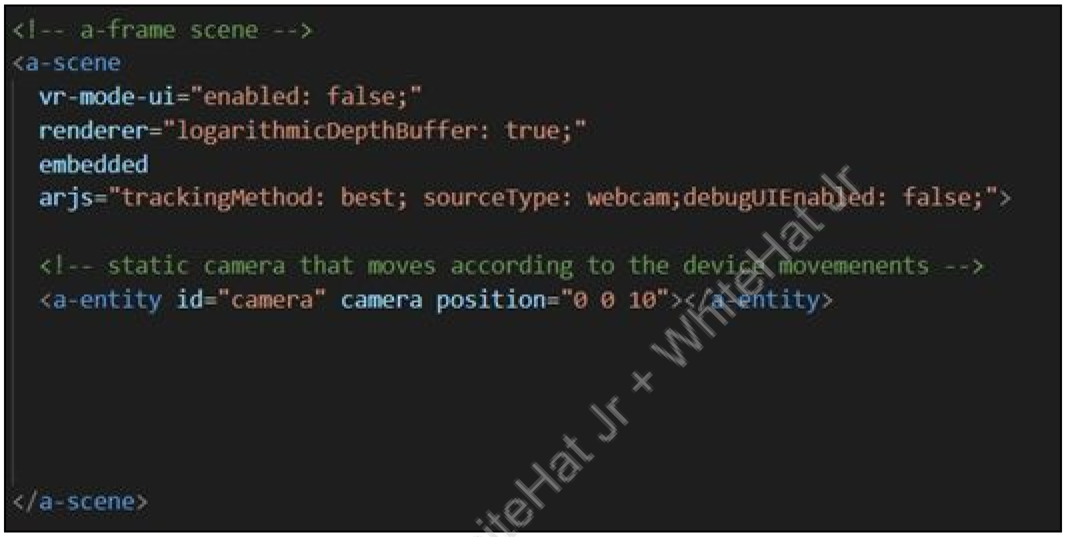
```
<!-- minimal loader shown until image descriptors are loaded.
  Loading may take a while according to the device computational power -->

<div class="arjs-loader">
  <div>Loading, please wait...</div>
</div>
```



```
<!-- style for the loader -->
<style>
  .arjs-loader {
    height: 100%;
    width: 100%;
    position: absolute;
    top: 0;
    left: 0;
    background-color: rgba(0, 0, 0, 0.8);
    z-index: 9999;
    display: flex;
    justify-content: center;
    align-items: center;
  }

  .arjs-loader div {
    text-align: center;
    font-size: 1.25em;
    color: white;
  }
</style>
```

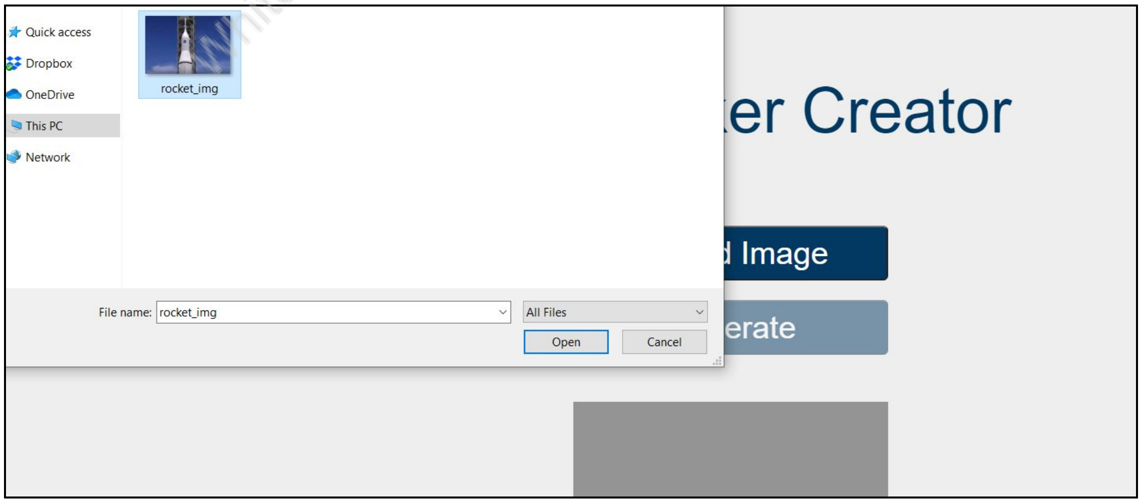
3. Set the <a-scene> element component:

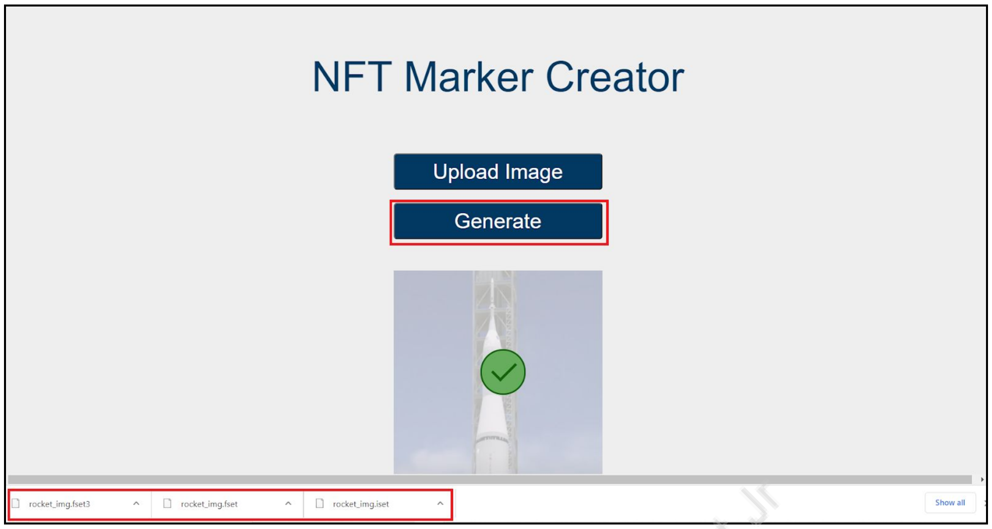
   vr-mode-ui="enabled: false;"
   - renderer="logarithmicDepthBuffer: true;"
   - embedded
   - arjs="tackingMethod: best; sourceType: webcam; debugUIEnable: false;"



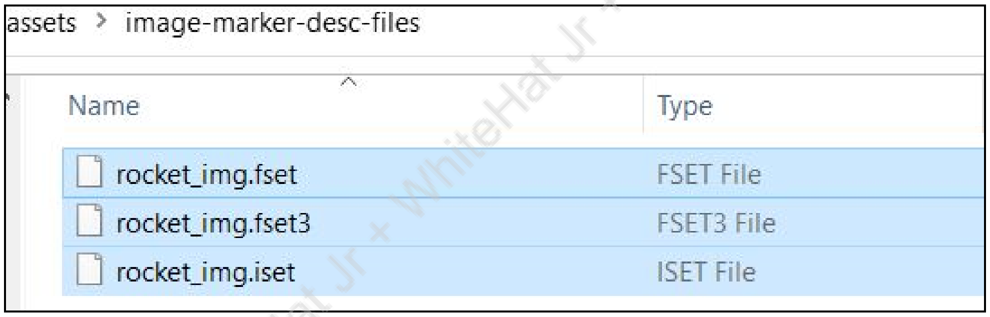
```
<!-- a-frame scene -->
<a-scene
  vr-mode-ui="enabled: false;"
  renderer="logarithmicDepthBuffer: true;"
  embedded
  arjs="trackingMethod: best; sourceType: webcam;debugUIEnabled: false;">

  <!-- static camera that moves according to the device movemenents -->
  <a-entity id="camera" camera position="0 0 10"></a-entity>




</a-scene>
```

4. Take an image and open it from the system, upload in the NFT to the creator, and click on "Generate".





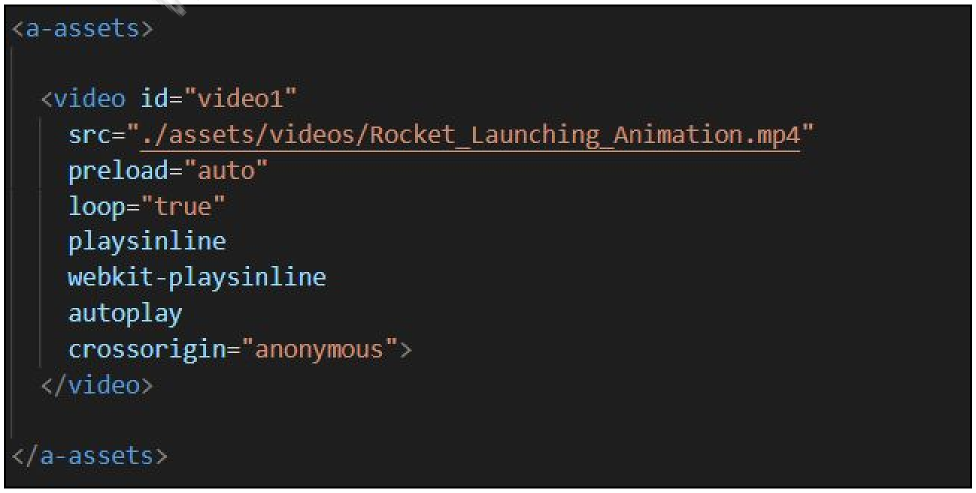
5. Check for the downloaded files and paste them into the working directory.



assets > image-marker-desc-files

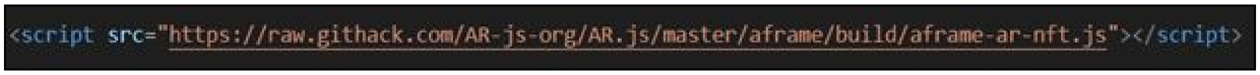
Name	Type
rocket_img.fset	FSET File
rocket_img.fset3	FSET3 File
rocket_img.iset	ISET File

6. Use <video> to add the video src files and set other properties to play the video.



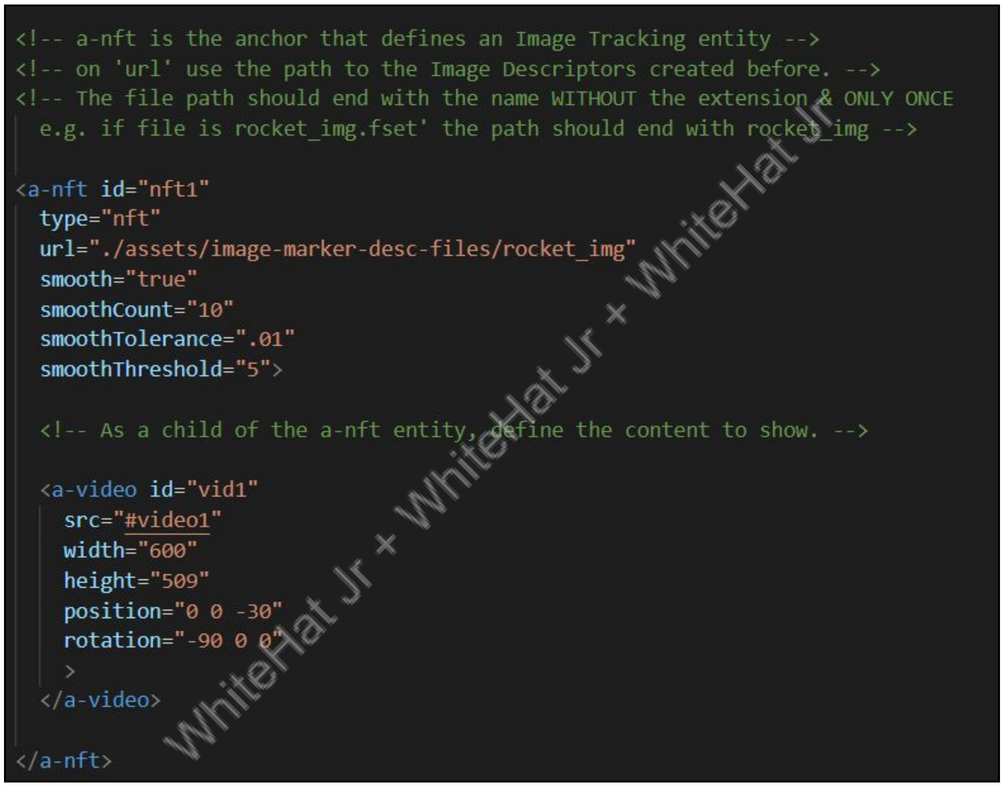
```
<a-assets>

  <video id="video1"
    src="./assets/videos/Rocket_Launching_Animation.mp4"
    preload="auto"
    loop="true"
    playsinline
    webkit-playsinline
    autoplay
    crossorigin="anonymous">
  </video>

</a-assets>
```

7. Add the aframe-ar-nft.js library and use the <a-nft> tag to add the nft marker files.



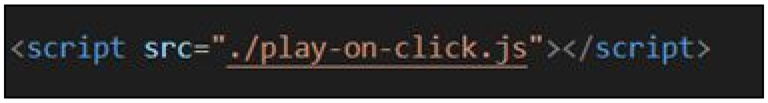
```html
<script src="https://raw.githack.com/AR-js-org/AR.js/master/aframe/build/aframe-ar-nft.js"></script>
```

8. To set the video entity, we will use <a-video> as the child of the <a-nft> and set the src id, height, width, position and rotation to set its orientation.

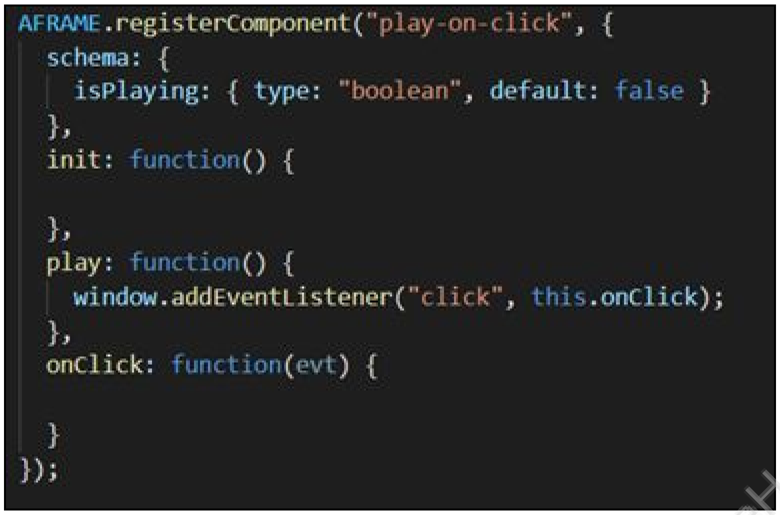


```html
<!-- a-nft is the anchor that defines an Image Tracking entity -->
<!-- on 'url' use the path to the Image Descriptors created before. -->
<!-- The file path should end with the name WITHOUT the extension & ONLY ONCE
  e.g. if file is rocket_img.fset' the path should end with rocket_img -->

<a-nft id="nft1"
  type="nft"
  url="./assets/image-marker-desc-files/rocket_img"
  smooth="true"
  smoothCount="10"
  smoothTolerance=".01"
  smoothThreshold="5">

  <!-- As a child of the a-nft entity, define the content to show. -->

  <a-video id="vid1"
    src="#video1"
    width="600"
    height="509"
    position="0 0 -30"
    rotation="-90 0 0"
    >
  </a-video>

</a-nft>
```

9. Register “play-on-click” components and add the schema & .init(), play() and onClick() functions.

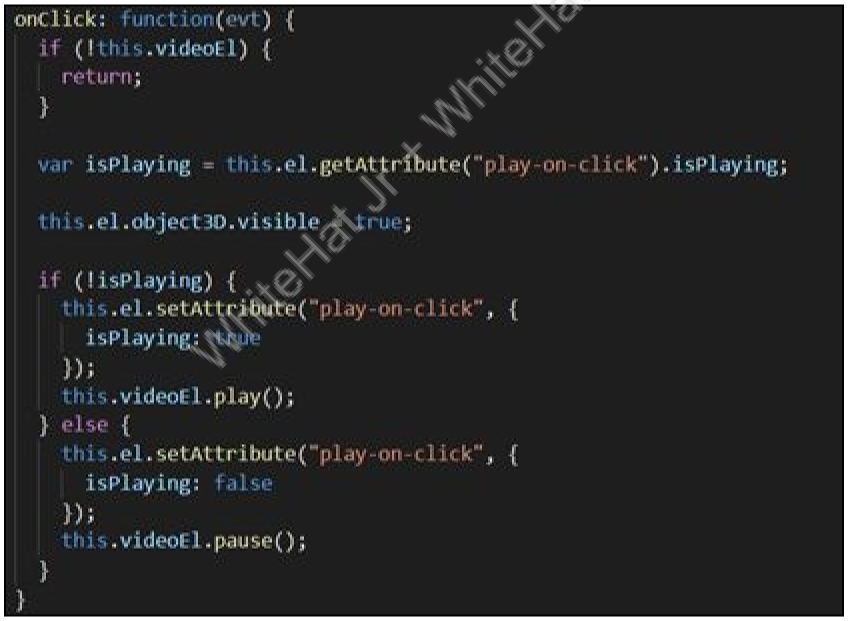


```html
<script src="./play-on-click.js"></script>
```

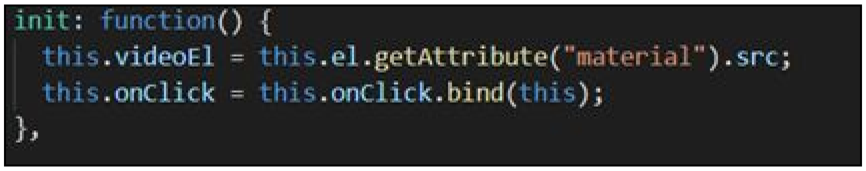


```
AFRAME.registerComponent("play-on-click", {
  schema: {
    isPlaying: { type: "boolean", default: false }
  },
  init: function() {

  },
  play: function() {
    window.addEventListener("click", this.onClick);
  },
  onClick: function(evt) {

  }
});
```

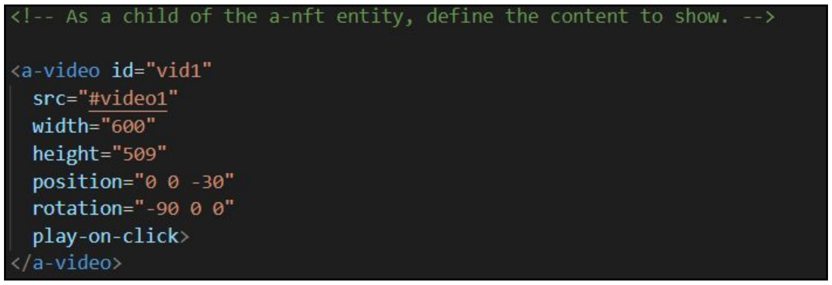
10. Write onClick() function in the component and take the **videoEl** variable and select the video src to be played on click in .init() method.



```
onClick: function(evt) {
  if (!this.videoEl) {
    return;
  }

  var isPlaying = this.el.getAttribute("play-on-click").isPlaying;

  this.el.object3D.visible = true;

  if (!isPlaying) {
    this.el.setAttribute("play-on-click", {
      isPlaying: true
    });
    this.videoEl.play();
  } else {
    this.el.setAttribute("play-on-click", {
      isPlaying: false
    });
    this.videoEl.pause();
  }
}
```

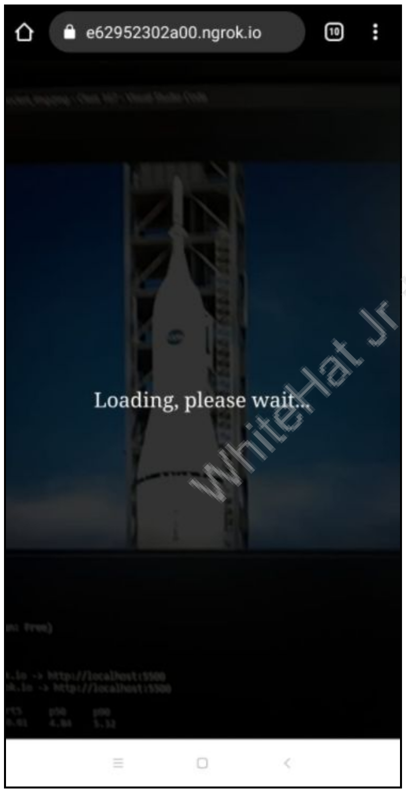


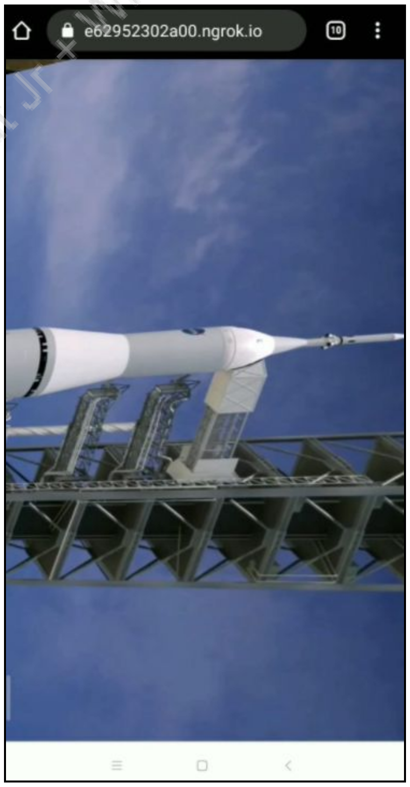
```
init: function() {
  this.videoEl = this.el.getAttribute("material").src;
  this.onClick = this.onClick.bind(this);
},
```



```
<!-- As a child of the a-nft entity, define the content to show. -->

<a-video id="vid1"
  src="#video1"
  width="600"
  height="509"
  position="0 0 -30"
  rotation="-90 0 0"
  play-on-click>
</a-video>
```

11. To see the output:
    - Use **ngrok** to run the application.
    - Open **HTTPS URL** in your smartphone/laptop & give permission to use the camera.
    - Open the **original image** that was used to create the nft image marker and point the camera towards it.





We have successfully learned to create the image tracker based web AR application to play the video content using that as a marker.

What's NEXT?

In the next class, we will learn about pattern marker based AR.

EXTEND YOUR KNOWLEDGE:

- You can refer to the link below to explore more about AFrame
  A-Frame
- You can refer to the link below to explore more about AR.js
  AR.js