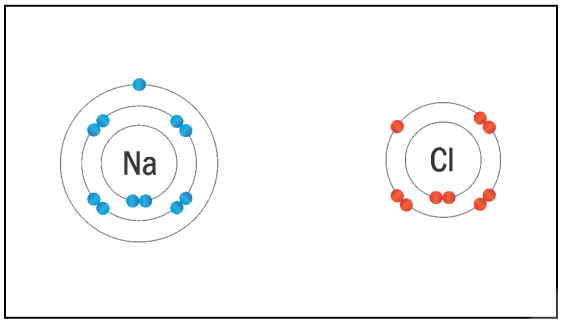
Topic	AR CHEMICAL COMPOUNDS	
Class Description	**Students will learn to find the distance between two markers. Students will also learn to show the chemical compound formation in Augmented reality based on barcode markers.**	
Class	**C175**	
Class time	**45 mins**	
Goal	• Learn to find the distance between two markers present in the Augmented reality of A-Frame. • Learn to show the compound formation of two or more atomic molecules when markers are at a certain distance in the Augmented reality scene.	
Resources Required	• Teacher Resources: ○ Visual Studio Code Editor ○ laptop with internet connectivity ○ smartphone ○ earphones with mic ○ notebook and pen • Student Resources: ○ Visual Studio Code Editor ○ laptop with internet connectivity ○ smartphone ○ earphones with mic ○ notebook and pen	
Class structure	**Warm-Up** **Teacher-led Activity** **Student-led Activity** **Wrap-Up**	**5 mins** **15 mins** **20 mins** **5 mins**
• **WARM UP SESSION - 5 mins**		

<table>
<tr><td colspan="3"><u>CONTEXT</u>
• Barcode marker based AR in A-Frame.</td></tr>
<tr><td colspan="3">Teacher Starts Slideshow 
Slide 1 to 4
Refer to speaker notes and follow the instructions on each slide.</td></tr>
<tr><td colspan="2">Hey <student's name>. How are you? It’s great to see you!
Are you excited to learn something new today?

Following are the WARM-UP session deliverables:
• Greet the student.
• Revision of previous class activities.
• Quizzes.</td><td>ESR: Hi, thanks!
Yes I am excited about it!

Click on the slide show tab and present the slides</td></tr>
<tr><td colspan="3">WARM-UP QUIZ
Click on In-Class Quiz</td></tr>
<tr><td colspan="3">Continue WARM-UP Session 
Slide 5 to 12</td></tr>
<tr><td colspan="3">Following are the session deliverables:
• Appreciate the student.
• Narrate the story by using hand gestures and voice modulation methods to bring in more interest in students.</td></tr>
<tr><td>Class Steps</td><td>Teacher Action</td><td>Student Action</td></tr>
<tr><td>Step 1:
Warm-Up
(5 mins)</td><td>We have many barcode markers for augmented reality and each one of them has a unique value assigned for identification.</td><td></td></tr>
</table>

	Today we are going to learn about the chemical compound formation between two elements in augmented reality. In the previous class we saw the atomic structure of the elements of the periodic table. I hope you read a little bit about how the compounds can be formed when two or more atoms combine together? Let's first understand how the compounds are formed, to be specific how **ionic compounds** are formed. There are a lot of ways how compounds can be formed but we will be just focusing on the ionic compound formation. Basically these compounds are formed by elements to become more stable. In very simple terms, to become stable **elements, elements which have one or more extra electrons to lose combine with the elements which will need the same number of electrons,** and that's how the **ionic compounds are formed**.	**ESR**: Yes

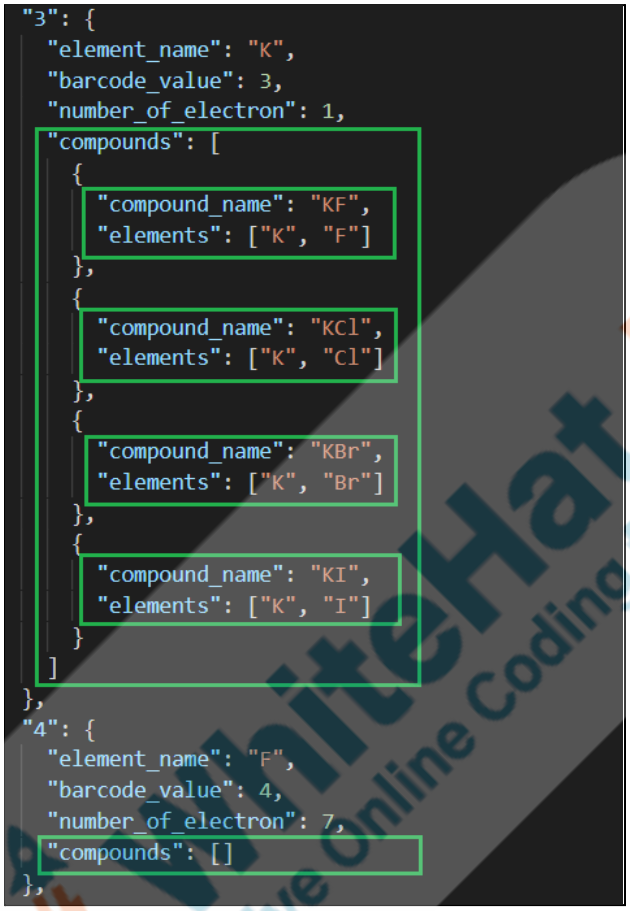
	 ***Note**: Teacher can show the GIF to the student for better understanding.* Now we have a little background of the formation of a compound, shall we begin? Are you excited?	**ESR**: Yes. **ESR**: Yes.
	Let's get started then.	
Teacher Ends Slideshow		
TEACHER-LED ACTIVITY - 15 mins		
Teacher Initiates Screen Share		
CHALLENGE • **Use Barcode Markers for Augmented reality.** • **Make Augmented reality chemical compounds in A-Frame.**		

Step 2: Teacher-led Activity (15 mins)	*<The teacher clones the code from Teacher Activity 1.>* ***[Teacher Activity 1]*** In the previous class, we created an AR atomic structure for the Na (Sodium) and Cl (Chloride) elements over two different barcode markers with unique values. These two elements can form a compound called NaCl (sodium chloride). Now today we want to show if the compound can be formed between two or more elements whose barcode markers are visible in the AR scene when we keep these markers very close to each other. ***Note**: Check out the output reference here for more clarity to explain the student in a better way.* Well this seems to be a bigger problem statement that we want to solve. To solve this we should break it down into smaller problems, right? ***Note**: Encourage the student to be more involved with you in finding the solutions and let the student come up with their own solutions.*	**ESR**: Yes.

	Can you help me with that? Superb! When we will show the compound, the atoms of the elements will disappear and only the compound will be visible. But we also need to know which of these two elements can form compounds. How would a computer understand that a compound can be formed between the two elements which are visible in the scene? Well, yes! That's what we are exactly going to do! We are going to give some information to the computer about which two elements can form compounds and then program it to show what we need. This is the beauty of programming! We can do a lot of unimaginable things with it.	**ESR**: ● First, we should find the distance between two markers to know if they are next to each other. ● If they are close enough, we need to show the compound over the marker. **ESR**: We can tell the computer which two elements can form compounds.

	Where should we keep this data to tell the computer which two elements can form the compounds? Yes. Perfect! *<The teacher opens the compoundList.json and explains about the "compounds".>* We will keep an array of **compounds** which will have the **compound_name** and the **elements** involved in the compound formation. We will do this for those elements only whose barcode marker value will show the compound after the markers are touching each other. We will keep the **array of compounds empty** for elements whose barcode marker will not be used to show the compound. ***Note**: Check out the output reference here for more clarity to explain the student in a better way.*	**ESR**: We should use a JSON file like we did before.

***Note**: The below code is already a part of the boilerplate.*



```
"3": {
  "element_name": "K",
  "barcode_value": 3,
  "number_of_electron": 1,
  "compounds": [
    {
      "compound_name": "KF",
      "elements": ["K", "F"]
    },
    {
      "compound_name": "KCl",
      "elements": ["K", "Cl"]
    },
    {
      "compound_name": "KBr",
      "elements": ["K", "Br"]
    },
    {
      "compound_name": "KI",
      "elements": ["K", "I"]
    }
  ]
},
"4": {
  "element_name": "F",
  "barcode_value": 4,
  "number_of_electron": 7,
  "compounds": []
},
```

	<The teacher opens Compounds.js and explains about adding compound nucleus, name and the base card in the AR scene.> Once the markers are close to each other, we will show the compound name and nucleus (sphere entity) of all elements of the compound attached together and it's base card. ***Note**: Check out the output reference here for more clarity to explain the student in a better way.*	

***Note**: The below code is already a part of the boilerplate.*

```
// adding compounds
var compounds = element.compounds;
compounds.map(item => {
  var compound = document.createElement("a-entity");
  compound.setAttribute("id", `${item.compound_name}-${barcodeValue}`);
  compound.setAttribute("visible", false);
  marker.appendChild(compound);

  var compoundCard = document.createElement("a-entity");
  compoundCard.setAttribute("id", `compound-card-${item.compound_name}`);
  compoundCard.setAttribute("geometry", {
    primitive: "plane",
    width: 1.2,
    height: 1.7
  });

  compoundCard.setAttribute("material", {
    src: `./assets/compound_cards/card_${item.compound_name}.png`
  });
  compoundCard.setAttribute("position", { x: 0, y: 0, z: 0.2 });
  compoundCard.setAttribute("rotation", { x: -90, y: 0, z: 0 });

  compound.appendChild(compoundCard);

  var posX = 0;
  item.elements.map((m, index) => {
    var n = document.createElement("a-entity");
    n.setAttribute("id", `compound-nucleus-${m}`);
    n.setAttribute("geometry", {
      primitive: "sphere",
      radius: 0.2
    });
    n.setAttribute("material", "color", colors[m]);
    n.setAttribute("position", { x: posX, y: 1, z: 0 });
    posX += 0.35;

    compound.appendChild(n);

    var nuclesName = document.createElement("a-entity");
    nuclesName.setAttribute("id", `compound-nucleus-name-${m}`);
    nuclesName.setAttribute("position", { x: 0, y: 0.21, z: 0 });
    nuclesName.setAttribute("rotation", { x: -90, y: 0, z: 0 });
    nuclesName.setAttribute("text", {
      font: "monoid",
      width: 3,
      color: "black",
      align: "center",
      value: m
    });

    n.appendChild(nuclesName);
  });
```

	<The teacher opens markerhandler.js file and explains it to the student.> Now we are going to write functions in the **markerhandler** component attached to the marker entity (created for each element in Compund.js in the previous class): • **.init()** to handle **markerFound** and **markerLost e**vents for all the barcode markers visible in the AR scene. • **.tick()** to call **showCompound()** function when the markers are at a certain distance. • **getCompound()** to return compounds (with name and barcode value) in the elements array. • **getDistance()** to calculate distance between any two position vectors which will help us to find the distance between markers.	

***Note**: The below code is already a part of the boilerplate.*

```
//add marker
var marker = document.createElement("a-marker");

marker.setAttribute("id", `marker-${barcodeValue}`);
marker.setAttribute("type", "barcode");
marker.setAttribute("element_name", elementName);
marker.setAttribute("value", barcodeValue);
marker.setAttribute("markerhandler", {});

scene.appendChild(marker);
```

***Note**: The below code is already a part of the boilerplate.*

```
AFRAME.registerComponent("markerhandler", {

  init: async function () {
    var compounds = await this.getCompounds();

    //Handle marker events
    this.el.addEventListener("markerFound", () => {


    });

    this.el.addEventListener("markerLost", () => {

    });

  },
  //Show compund based on the distance
  tick: function () {

  },

  //Calculate distance between two position vectors
  getDistance: function (elA, elB) {

  },

  //Add elements to array whch can form compound
  getCompound: function () {

  },
  getCompounds: function () {
    return fetch("js/compoundList.json")
      .then(res => res.json())
      .then(data => data);
  },
  //Function to show Compound
  showCompound: function (compound) {
    elementsArray.map(item => {
      var el = document.querySelector(
        `#${item.element_name}-${item.barcode_value}`
      );
      el.setAttribute("visible", false);
    });
    var compound = document.querySelector(
      `#${compound.name}-${compound.value}`
    );
    compound.setAttribute("visible", true);
  },
});
```

	We will start with handling the **markerFound** event. Can you tell me when this event is triggered? Yes. Perfect! So, whenever the marker is found: • For any element in the scene, we should display the atomic structure (created in the previous class) on their respective barcode markers. Hence, the **visibility attribute of the atom entity** (having the nucleus, electrons and orbits as the child attribute) **should be set to true**. • But the final compound entity (the compound name, sphere entity of elements forming the compounds and it's base card) that we have to show **should not be visible at that time**. • We will keep a global array variable, called **elementsArray**, to keep track of all the elements which can form compounds together. `var elementsArray = [];`	**ESR**: Whenever we have any marker like Hiro marker, pattern marker or barcode in the augmented reality scene.

	Whenever the marker is visible in the scene for any element, we will push that element into an array. *<The teacher opens markerhandler.js file and updates **markerFound** event in the .init() function.>*	

```
this.el.addEventListener("markerFound", () => {
  var elementName = this.el.getAttribute("element_name");
  var barcodeValue = this.el.getAttribute("value");
  elementsArray.push({ element_name: elementName, barcode_value: barcodeValue });

  // Changing Compound Visiblity
  compounds[barcodeValue]["compounds"].map(item => {
    var compound = document.querySelector(`#${item.compound_name}-${barcodeValue}`);
    compound.setAttribute("visible", false);
  });

  // Changing atom Visiblity
  var atom = document.querySelector(`#${elementName}-${barcodeValue}`);
  atom.setAttribute("visible", true);
});
```

	And if the marker of any element is lost from the scene we will remove that particular element from **elementsArray**. We will use the **findIndex()** function to find the index of the element whose marker is lost from the scene and use the **slice()** function to remove that element from the array. *<The teacher opens markerhandler.js file and updates **markerFound** event in the .init() function.>*	

```
this.el.addEventListener("markerLost", () => {
  var elementName = this.el.getAttribute("element_name");
  var index = elementsArray.findIndex(x => x.element_name === elementName);
  if (index > -1) {
    elementsArray.splice(index, 1);
  }
});
```

Now we will take **two arrays** (**A** & **B**) for compounds which can be formed with just two elements (like NaCl, KCl):

- One with elements that can lose electrons to form compounds that are elements like **Na** (Sodium), **K** (Potassium), **Li** (Lithium) and **H** (Hydrogen).
- And another one with elements that need electrons to form compounds that are elements like **F** (Fluorine), **Cl** (Chlorine), **Br** (Bromine) and **I** (Iodine).

```
var A = ["H", "Li", "Na", "K"];
var B = ["F", "Cl", "Br", "I"];
```

For now we are restricting elements of **A** to form compounds with elements **B** only.

Now we will write a function **getCompund()** which will return the name of the compound and the barcode value of the marker which will show the compound:

- Loop through **elementsArray:**
    - Check if **A** has the element_name.
    - Add the element_name to a variable.
    - Loop through **elementsArray:**
        - Check if **B** has the element_name.
        - Join the element_name to the same variable.
        - Return the name and barcode value of the first element.

```
getCompound: function () {
  for (var el of elementsArray) {
    if (A.includes(el.element_name)) {
      var compound = el.element_name;
      for (var i of elementsArray) {
        if (B.includes(i.element_name)) {
          compound += i.element_name;
          return { name: compound, value: el.barcode_value };
        }
      }
    }
  }
},
```

Now once we have the compound name and barcode marker value on which we

want to show the compound, we can now **calculate the distance**.

For this we are going to use the Three.js method **distanceTo()**.

For example **elA.distanceTo(elB)** will give distance between elA and elB, where elA and elB are position vectors as Three.js objects.

```
getDistance: function (elA, elB) {
  return elA.object3D.position.distanceTo(elB.object3D.position);
},
```

Now we will call this function(**getDistance()**) to get the distance between two barcode markers:

If it can be formed then we'll show the compound using all

If the compound cannot be formed then we'll show the text message-"**Compound will not form…**"

Can you tell how we should write this?

To do this we will write inside the **.tick()** function of the component. The .tick() functions is called continuously:

- Check if **elementsArray** length is greater than zero.
- Select "**message-text**" entity.
- Call **getCompund()**.

- If **elementsArray** length is equal to two:
    - Find distance between two markers using **getDistance()** function**.**
    - If distance is less than 1.25:
        - Check if the compound is defined and call **showCompund()**, which sets the visible attribute for compounds to true and for elements to false.
    - Else show **message-text.**

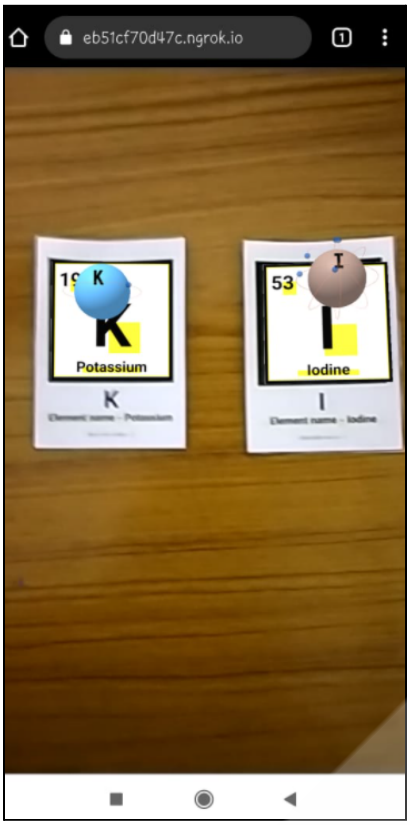
```
<a-plane
  id="message-text"
  position="0 0 -3.2"
  rotation="0 0 0"
  width="1.6"
  height="0.15"
  color="#FFFFFF"
  visible="false"
>
  <a-entity
    text="font:monoid;value:Compound will not form...;align:center;width:2.1;color:#c2185b;"
  ></a-entity>
</a-plane>
```

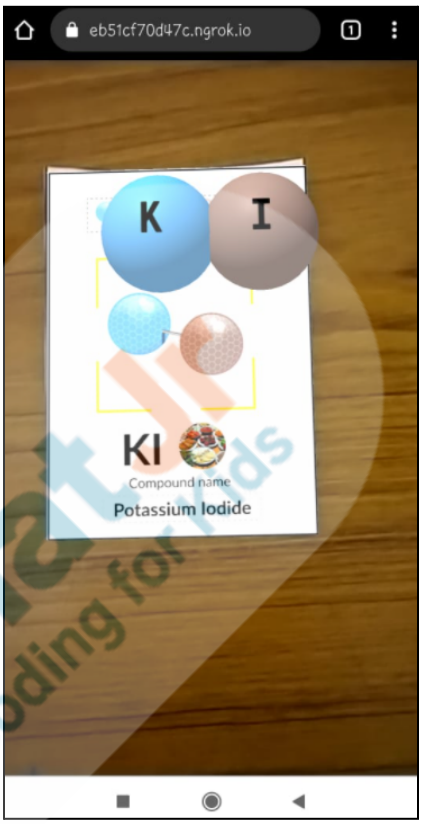
```
showCompound: function (compound) {
  elementsArray.map(item => {
    var el = document.querySelector(`#${item.element_name}-${item.barcode_value}`);
    el.setAttribute("visible", false);
  });

  // show Compound
  var compound = document.querySelector(`#${compound.name}-${compound.value}`);
  compound.setAttribute("visible", true);
},
```

```
tick: function () {
  if (elementsArray.length > 1) {

    var messageText = document.querySelector("#message-text");

    var length = elementsArray.length;
    var distance = null;

    var compound = this.getCompound();

    if (length === 2) {
      var marker1 = document.querySelector(`#marker-${elementsArray[0].barcode_value}`);
      var marker2 = document.querySelector(`#marker-${elementsArray[1].barcode_value}`);

      distance = this.getDistance(marker1, marker2);

      if (distance < 1.25) {
        if (compound !== undefined) {
          this.showCompound(compound);
        } else {
          messageText.setAttribute("visible", true);
        }
      } else {
        messageText.setAttribute("visible", false);
      }
    }
  }
},
```

Now we can test the output using ngrok.

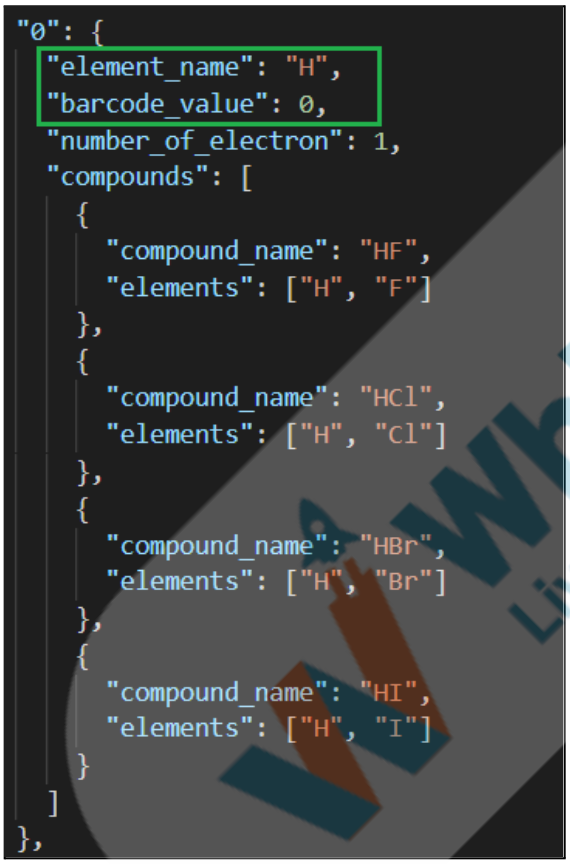
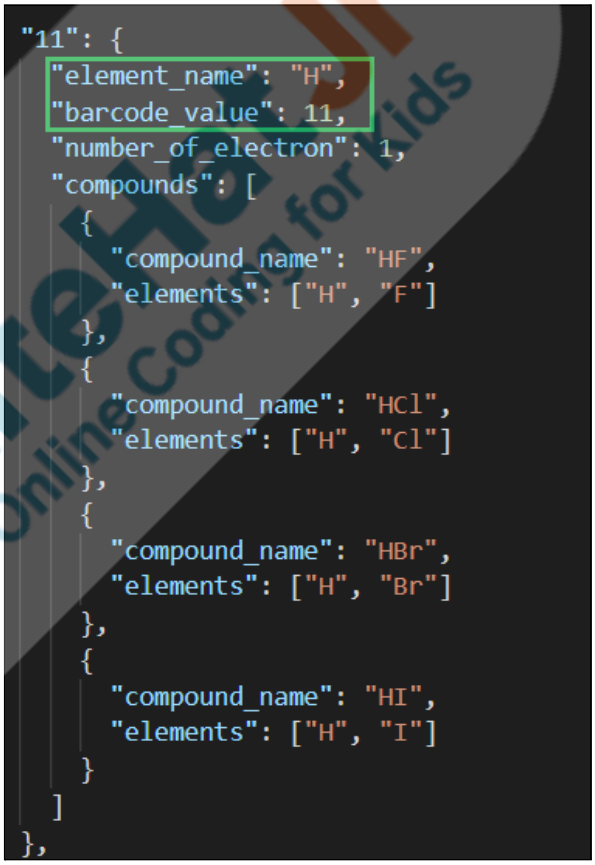
Note: *Scan (Point the camera towards) the marker image to see the output.*





	That's pretty interesting to find the distance between markers and form the compounds. We could do this for two elements in the scene. Now you will do this for compounds like water (H_2O) which needs three elements to form compounds. Are you excited?	**ESR**: Yes!
Teacher Stops Screen Share		
	Now it's your turn. Please share your screen with me.	

Teacher Starts Slideshow  **Slide 13 to 15** Refer to speaker notes and follow the instructions on each slide.		
We have one more class challenge for you. Can you solve it? Let's try. I will guide you through it.		
Teacher Ends Slideshow 		
STUDENT-LED ACTIVITY - 20 mins		
• **Ask the student to press the ESC key to come back to the panel.** • **Guide the student to start screen share.** • **Teacher gets into fullscreen.**		
ACTIVITY • **Show compound formation for three elements in augmented reality.**		
Step 3: Student-Led Activity (20 mins)	*The teacher guides the student to clone the code from Student Activity 1.* ***[Student Activity 1]*** ***Note**: The student will continue to add new functionality after teacher activity.*	
	Now we want to show compounds with three elements like H_2O, or K_2S. Let's pick up H_2O to start with. This element has two hydrogen elements and one oxygen element.	

	Since H_2O has two hydrogen elements, we will use two unique barcode id values for each one of them. *Explain the student about the elements data in the compoundList.json file.*	
		
	For elements like **O** (Oxygen), **S** (Sulphur) and **Se** (Selenium) we keep these in a different array, **C,** that needs two elements from the array **A** to show the compound formation. ***Note**: Elements of **A** can form*	

*compounds with elements **C**.*

```
var A = ["H", "Li", "Na", "K"];
var B = ["F", "Cl", "Br", "I"];

var C = ["O", "S", "Se"];

var elementsArray = [];
```

We will now write the function **countOccurrences()** to count the number of elements from array **A** that are there in the scene.

Can you tell me how we know if the elements are present in the scene?

ESR: If the barcode marker for that element is present in the AR scene in front of the camera.

Yes. Exactly!

We will use the **reduce()** function for JavaScript array that will take an **array** (elementsArray) and **value to be found** in that array (element_name) as the parameters to find out the occurrence of the element in the scene.

If found, we will add the element name into the array.

Guide the student to write the function.

```
countOccurrences: function (arr, val) {
  return arr.reduce((a, v) => (v.element_name === val ? a + 1 : a), 0);
},
```

	We will now call the function **countOccurrences()** in **getCompound()** which returns the final name of the compound and the barcode value for array **C**. *Guide the student to update the **getCompound()** function and add condition for elements present in array **C**.*	
	`` getCompound: function () { for (var el of elementsArray) { if (A.includes(el.element_name)) { var compound = el.element_name; for (var i of elementsArray) { if (B.includes(i.element_name)) { compound += i.element_name; return { name: compound, value: el.barcode_value }; } if (C.includes(i.element_name)) { var count = this.countOccurrences(elementsArray, el.element_name); if (count > 1) { compound += count + i.element_name; return { name: compound, value: i.barcode_value }; } } } } } }, ```	
	Now update the **.tick()** function to show compounds for three elements: • Check if **elementsArray** length is equal to three: ○ Find the distance between marker1 and marker2 & marker1 and marker3	

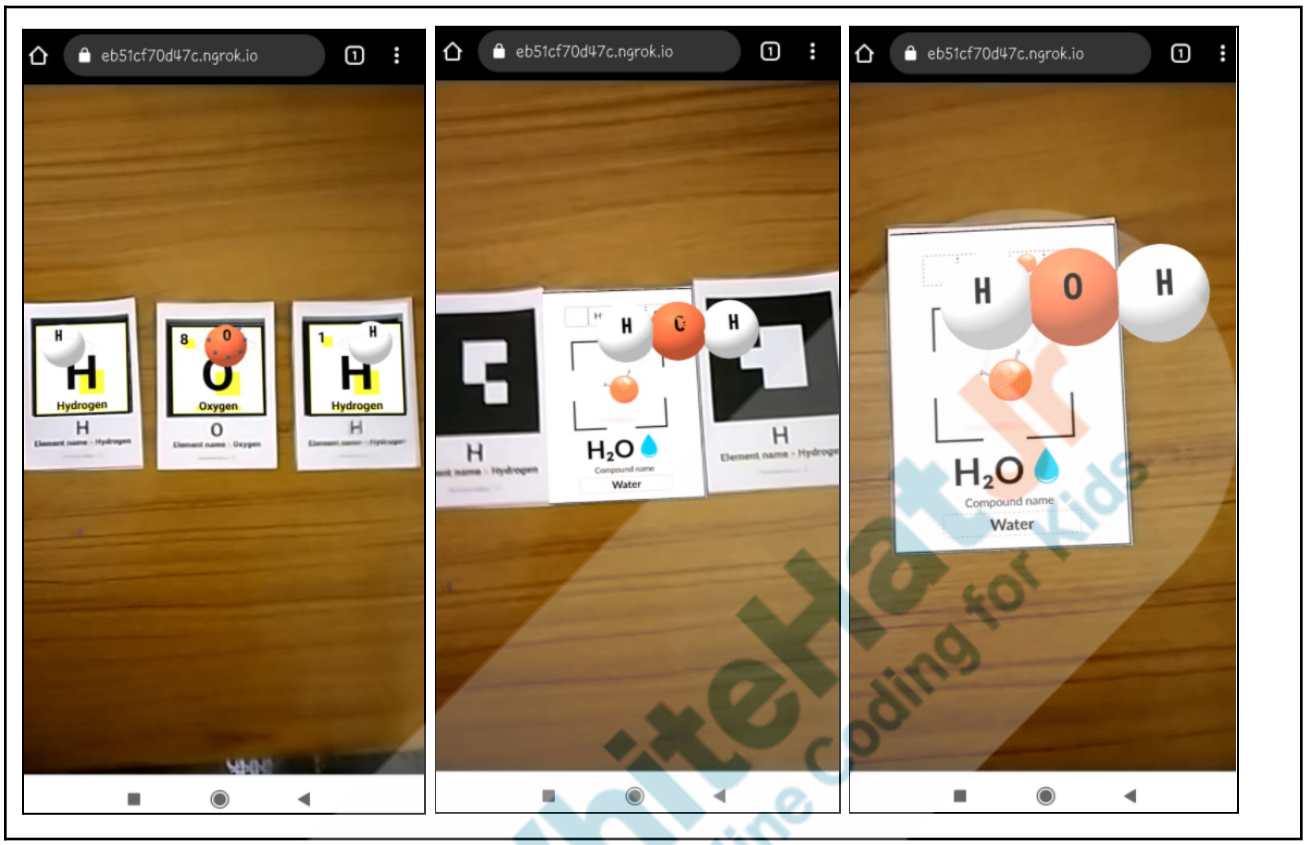
	using **getDistance()** function. ○ If distance is less than 1.25: ■ Check if the compound is defined and call **showCompund()**, which sets the visible attribute for compounds to true and for elements to false. ○ Else show **message-text.** *Guide the student to update the .tick() function and the conditions to show compound of the three elements when the three markers are touching each other.*	

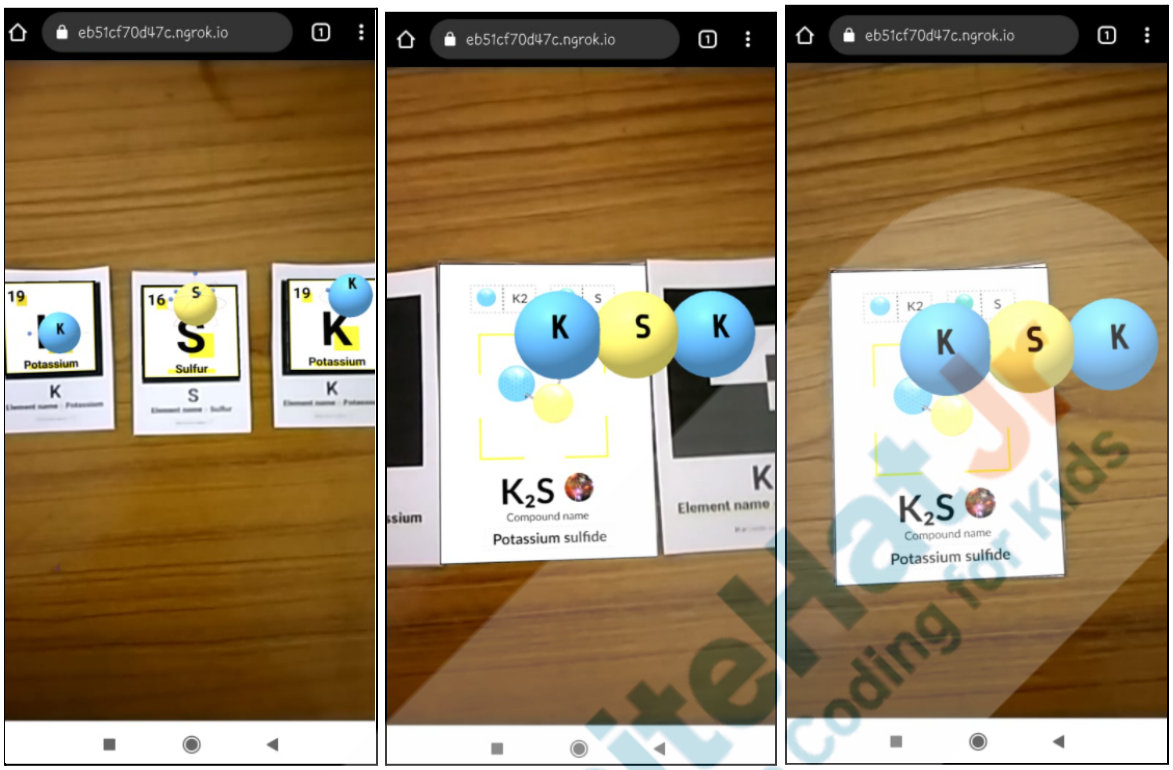
```
if (length === 3) {
  var marker1 = document.querySelector(`#marker-${elementsArray[0].barcode_value}`);

  var marker2 = document.querySelector(`#marker-${elementsArray[1].barcode_value}`);

  var marker3 = document.querySelector(`#marker-${elementsArray[2].barcode_value}`);

  var distance1 = this.getDistance(marker1, marker2);
  var distance2 = this.getDistance(marker1, marker3);

  if (distance1 < 1.25 && distance2 < 1.25) {

    if (compound !== undefined) {
      var barcodeValue = elementsArray[0].barcode_value;
      this.showCompound(compound, barcodeValue);
    } else {
      messageText.setAttribute("visible", true);
    }
  }
  else {
    messageText.setAttribute("visible", false);
  }
}
```

Guide the student to test the output using ngrok.

***Note 1**: It would be better to take the printout of these marker images to see the output better.*

***Note 2**: Encourage the student to take the printout of the marker images and keep them ready by cutting each one along the borders.*





Teacher Guides Student to Stop Screen Share

- **WRAP UP SESSION - 5 mins**

Teacher Starts Slideshow
Slide 16 to 19

Activity details

Following are the WRAP-UP session deliverables:

- Appreciate the student.
- Revise the current class activities.
- Discuss the quizzes.

WRAP-UP QUIZ
Click on In-Class Quiz

Continue WRAP-UP Session

Slide 20 to 25		
Activity Details **Following are the session deliverables:** • Explain the facts and trivia • Next class challenge • Project for the day • Additional Activity (Optional)		
FEEDBACK • **Compliment the student for her/his effort in the class.** • **Encourage the student to think and come up with their own solutions.**		
You get a “hats-off”. Alright. See you in the next class.		• *Make sure you have given at least 2 Hats Off during the class for:* •  •  • 
PROJECT OVERVIEW DISCUSSION Refer the document below in Activity Links Sections		
Teacher Clicks 		
Additional Activities	*Encourage the student to write reflection notes in their reflection journal using markdown.*	*The student uses the markdown editor to write their reflections in a reflection journal.*

	Use these as guiding questions: ● What happened today? ○ Describe what happened. ○ The code I wrote. ● How did I feel after the class? ● What have I learned about programming and developing games? ● What aspects of the class helped me? What did I find difficult?	

Activity	Activity Name	Links
Teacher Activity 1	Boilerplate Code	https://github.com/whitehatjr/PRO-C175-Teacher-Boilerplate
Teacher Activity 2	Teacher Reference Code	https://github.com/whitehatjr/PRO-C175
Teacher Activity 3	Ionic Bond Example GIF	https://obj.whitehatjr.com/4044eb28-3837-441e-8f52-477682fbb5b9.gif
Teacher Activity 4	Output Reference	https://curriculum.whitehatjr.com/PRO+Asset/PRO+175+Output+Ref-AR+Chemistry.mp4
Student Activity 1	Boilerplate Code	https://github.com/whitehatjr/PRO-C175-Student-Boilerplate
Teacher Reference 1	Ngrok Updates	https://docs.google.com/document/d/1dlMry188llEJl6rHEc3AkBashQSOwGQ40HQft29S8vQ/edit?usp=sharing
Teacher Reference 2	Project Document	https://s3-whjr-curriculum-uploads.whjr.online/27dd21d0-5e6f-4b38-a041-24d27f1eeda8.pdf
Teacher	Project Solution	

Reference 3		https://github.com/whitehatjr/PRO-C175-AR
Teacher Reference 4	Visual-Aid	https://s3-whjr-curriculum-uploads.whjr.online/4233978f-d3f2-44c8-813e-151a0eaeb4e5.html
Teacher Reference 5	In-Class Quiz	https://s3-whjr-curriculum-uploads.whjr.online/7a04f62e-4e1d-4161-b4d3-215d979eb5ef.pdf