


Topic	STRUCTURING BEFORE CODING	
Class Description	Students design a form using p5 dom to allow players to log in. Students use the OOPs programming style to write the code.	
Class	C36	
Class time	45 mins	
Goal	<ul style="list-style-type: none"> • Create a form to log the players' names in the game. • Create a Database Structure and connect with the game. • Use OOPs programming style. 	
Resources Required	<ul style="list-style-type: none"> • Teacher Resources <ul style="list-style-type: none"> ○ VS Code Editor ○ Laptop with internet connectivity ○ Earphones with mic ○ Notebook and pen • Student Resources <ul style="list-style-type: none"> ○ VS Code Editor ○ Laptop with internet connectivity ○ Earphones with mic ○ Notebook and pen 	
Class structure	WARM-UP Teacher-led Activity Student-led Activity WRAP-UP	5 mins 10 min 20 min 5 mins
WARM-UP SESSION - 5 mins		
<div>  </div> <p>Teacher starts slideshow from slides 1 to 19</p> <p>Refer to speaker notes and follow the instructions on each slide.</p>		
Activity details		Solution/Guidelines
<i>How are you doing? Are you excited to learn something new?</i>		ESR: Thanks, yes I am excited about it.

Run the presentation from slides 1 to 4.

Click on the slide show tab and present the slides.

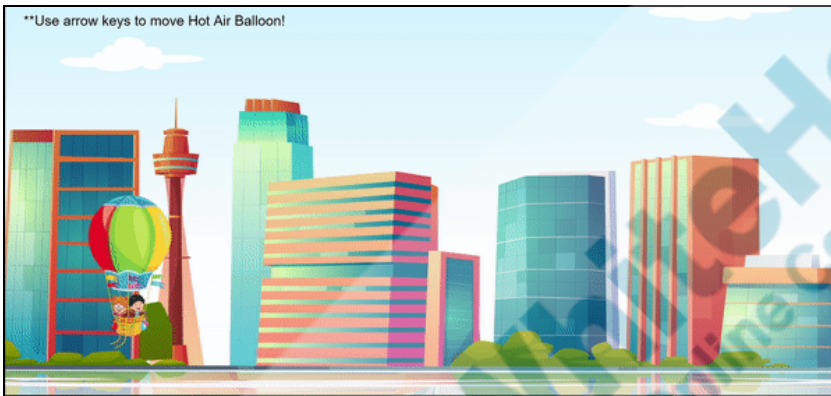
Following are the WARM-UP session deliverables:

- Connecting students to the previous class.
- WARM-UP Quiz Session.

QnA Session

Question

Select the correct block of code to write the function readHeight to read the height from the database.



```
function readHeight(data){
  balloon.x = height.x;
  balloon.y = height.y;
}
```

A.

```
function readHeight(data){
  height = data.val();
  balloon.x = height.x;
  balloon.y = height.y;
}
```

B.

```
function readHeight(data){
  height = data.val();
}
```

C.

Answer

B

<p>D.</p> <pre>function readHeight(){ height = val(); balloon.x = height.x; balloon.y = height.y; }</pre>	
<p>What will the following code block do?</p> <pre>function updateHeight(x,y){ database.ref('balloon/height').set({ 'x': height.x + x , 'y': height.y + y }) }</pre> <p>A. It is referring to the height node in the database and it is updating the width and height nodes.</p> <p>B. It is referring to the balloon node in the database and it is updating the width and height nodes.</p> <p>C. It is referring to the height node inside the balloon node in the database, and it is updating the x and y nodes.</p> <p>D. None of the above.</p>	<p>C</p>
<p>Continue the WARM-UP session</p>	
Activity details	Solution/Guidelines
<p>Run the presentation from 5 to 19 to set the problem statement.</p> <p>Following are the WARM-UP session deliverables:</p> <ul style="list-style-type: none"> • Appreciate the student on his/her performance in the quizzes. • Introduction to Code Structuring. • Creating a database. • Connect code to the database. 	<p><i>Narrate the story by using hand gestures and voice modulation methods to bring in more interest in students.</i></p>



Teacher ends slideshow

TEACHER-LED ACTIVITY - 10 mins

Teacher Initiates Screen Share

CHALLENGE

- Use p5 dom to create a login form for players to log in.

Activity details	Solution/Guidelines
<p>Let's say we make a 2-player game. When the number of registered players reaches 2, we want the game state to become 1 (PLAY). When the game state changes to 1, we would want the game to start.</p> <p>Any ideas on how to do this?</p>	<p>ESR: Varied</p> <p>ESR: varied</p>
<p>There are a number of ways in which we can go about doing this.</p> <p>We can start writing the code immediately. But good programmers, before writing code, think about how to structure their code.</p> <p>Which programming style are we using in our codes so far?</p>	<p>ESR: OOPs - object-oriented style</p>
<p>Let us find some answers to simple questions together like:</p> <p>How do we want players to join the game?</p> <p>Where and how do we save that information?</p>	<p>ESR: varied</p>

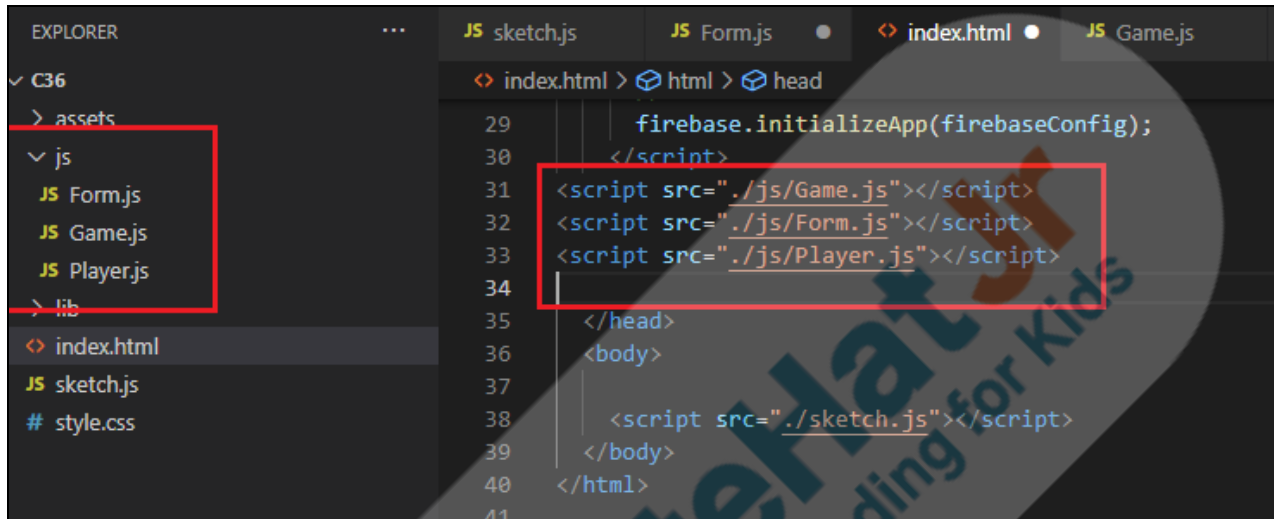
<p>You may have played games where you enter your details, how do you submit that information?</p> <p>Yes, such pages which ask for users to input information are called Forms.</p> <p>What are other objects that can be included in our game? What will their properties and functions be?</p>	<p><i>Prob the response like we need input box and a button to submit the name/ Bring the student to mention database to store player information/ You can give a hint as it was used in the last class.</i></p> <p>ESR: Varied</p>
<p>We need to have at least 3 objects:</p> <ol style="list-style-type: none"> 1. Form: The form should contain the input box and a button to log in. <ul style="list-style-type: none"> • When the button is pressed, the player's name should be registered in the database and a new player should be created. 2. Player: A new player object should be created every time a new player logs in. It should contain all the information about the player - name, position in the game, and so on. We can add multiple properties about the player as required by the game. 3. Game Object: The game object should be able to hold the state of the game. It should be able to display form when the game state is 0(WAIT), the game when the game state is 1(PLAY), and the leaderboard when the game state is 2(END). <ul style="list-style-type: none"> • To start, we will only consider the case when the game state is 0. 	<p><i>The student listens and asks questions.</i></p> <p><i>Prob student to think and speak about what information about the player will be needed in the game.</i></p>
<p>Now that we know how the basic structure of our program will be, it will become fairly easy to write our code!</p>	<p><i>The student shares the screen, fires up the editor, and prepares to code.</i></p>

<p>Without this structure, writing code can appear complex.</p> <p>With this guide in mind, why don't you start with the coding exercise? I will be guiding you in the exercise.</p> <p><i>Teacher opens Teacher Activity 1.</i></p> <p>Let us read sketch.js together. Can you explain to me what code is given here?</p> <p><i>[In case the student is getting stuck somewhere; the teacher should help the student and explain the code.]</i></p> <p><i>In case, the student is able to explain the code correctly then appreciate the student by saying:]</i></p> <p>Great job <student name> !!</p> <p>We also see one new function here;</p> <p>The windowResized() function in p5.js is called once every time the browser window is resized. It adjusts its height and width automatically whenever the size of the window is increased. This function is invoked automatically as soon as the window is resized and then creates a new canvas corresponding to it.</p>	<p>ESR:</p> <p>In sketch.js, various global variables are declared viz.</p> <pre>canvas; backgroundImage; database; form & player</pre> <ul style="list-style-type: none"> • The preload() function is used to load the background image. • In the setup() function, the canvas is created and the database is initialized. • The game object is created from the Game class and calls the start() function. • In the draw() function only the background command is given.
<pre>function windowResized() { resizeCanvas(windowWidth, windowHeight); }</pre>	
<p>As you can see the game has a js folder which contains all three object files which we discussed earlier.</p> <ul style="list-style-type: none"> • form.js 	<p><i>The student observes and listens.</i></p>

- game.js
- player.js

These files are added to index.html

[Teacher can quickly show index.html to the student.]



The screenshot shows a VS Code editor with the Explorer sidebar on the left. The 'js' folder is expanded, showing 'Form.js', 'Game.js', and 'Player.js'. The main editor shows 'index.html' with the following code:

```

29     firebase.initializeApp(firebaseConfig);
30     </script>
31     <script src="./js/Game.js"></script>
32     <script src="./js/Form.js"></script>
33     <script src="./js/Player.js"></script>
34
35 </head>
36 <body>
37
38     <script src="./js/sketch.js"></script>
39 </body>
40 </html>
41

```

Now, let us go through Game.js.

Guide the student to open Game.js and explain the code; help wherever needed.

The student opens Game.js and explains the code.

ESR:

The game class has no property in **start()** , which is being called from **sketch.js**; we are creating objects for class Form and class Player. We are also calling the **display()** function of the form.


```
class Game {  
  constructor() {}  
  
  start() {  
    form = new Form();  
    form.display();  
    player = new Player();  
  }  
}
```

Let's go to the Form Class now.

HTML is used to create any content like a form on a page.
HTML is similar to markdown in some ways.

An HTML contains elements that define the structure of a page. A simple html page contains:

- **head**: where all the scripts and stylesheets for the page are added.
- **body**: where all the content of the page is added.

The **body** of an HTML page can contain several different types of elements:

- **h1, h2, h3**: display headings of different sizes.
- **input**: to collect input from the user.
- **button**: to display a button.

This model of an HTML page is called the Document Object Model (or DOM).

We will be using the p5 Dom library to create the form.

You can look at the reference of the P5 dom library on how it is used.

[\(Teacher Activity 2\)](#)

The student listens and observes; may ask occasional questions.

The student observes and asks questions.

[Student Activity 2](#)


```
constructor() {
  this.input = createInput("").attribute("placeholder", "Enter your name");
  this.playButton = createButton("Play");
  this.titleImg = createImg("./assets/title.png", "game title");
  this.greeting = createElement("h2");
}
```

Each element is created in the **constructor()** in the Form class.

this.input, will display an empty box with the text given in the placeholder attribute.

The other element used is a button, which is clickable with the text "Play" on it.

We are also using **createImg**, which takes in "image path & label" as a parameter.

The last one is a simple text message, which will be displayed to welcome the player.

[Teacher should open the reference link [Teacher Activity 2](#) and show all elements from there.]

Let's write functions to assign display positions and styles to these elements.

And we will call them in a **display()** function.

The student observes and asks questions.

The student observes and asks questions

The teacher writes following methods in form.js

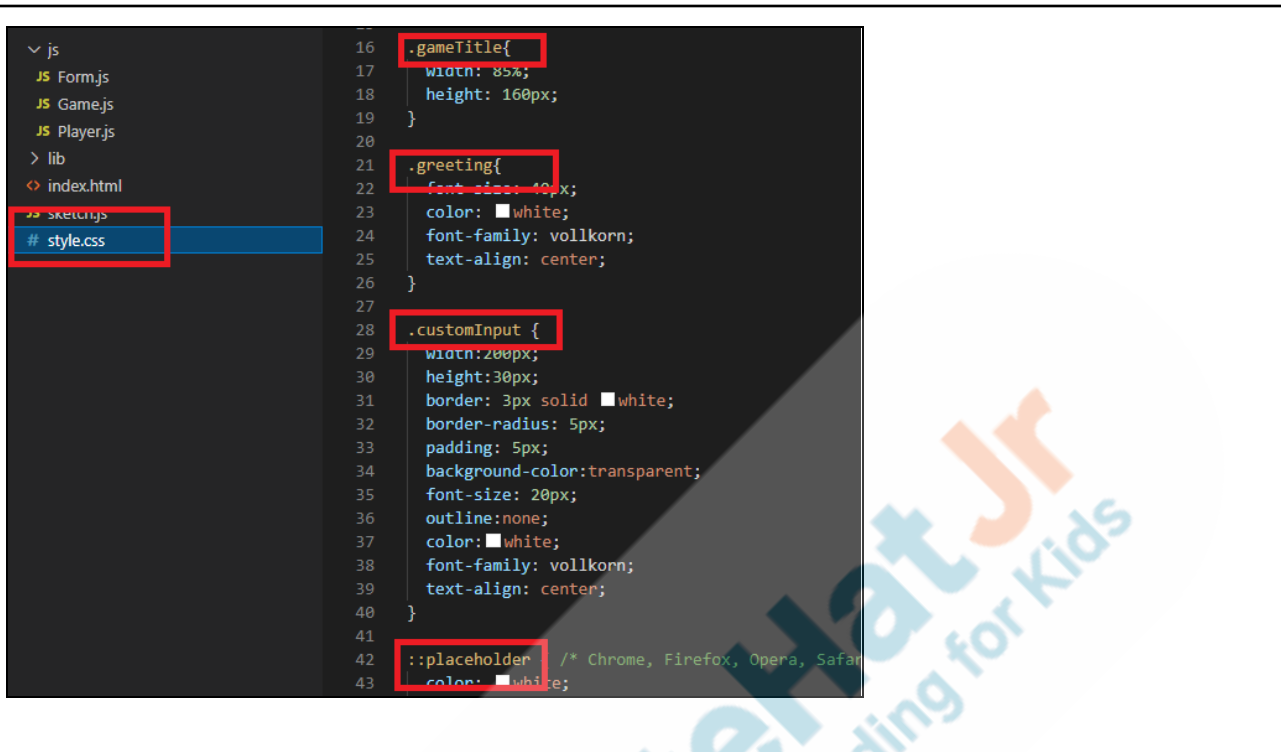
```
setElementsPosition() {  
  this.titleImg.position(120, 160);  
  this.input.position(width / 2 - 110, height / 2 - 80);  
  this.playButton.position(width / 2 - 90, height / 2 - 20);  
  this.greeting.position(width / 2 - 300, height / 2 - 100);  
}
```

Change X & Y position as per
Screen size. Try (120, 50)

```
setElementsStyle() {  
  this.titleImg.class("gameTitle");  
  this.input.class("customInput");  
  this.playButton.class("customButton");  
  this.greeting.class("greeting");  
}
```

```
display() {  
  this.setElementsPosition();  
  this.setElementsStyle();  
}
```

The function `setElementsStyle()` uses the styles defined in `style.css`; the teacher can show students a glimpse of it. The teacher can mention it will be explained in detail in upcoming classes.



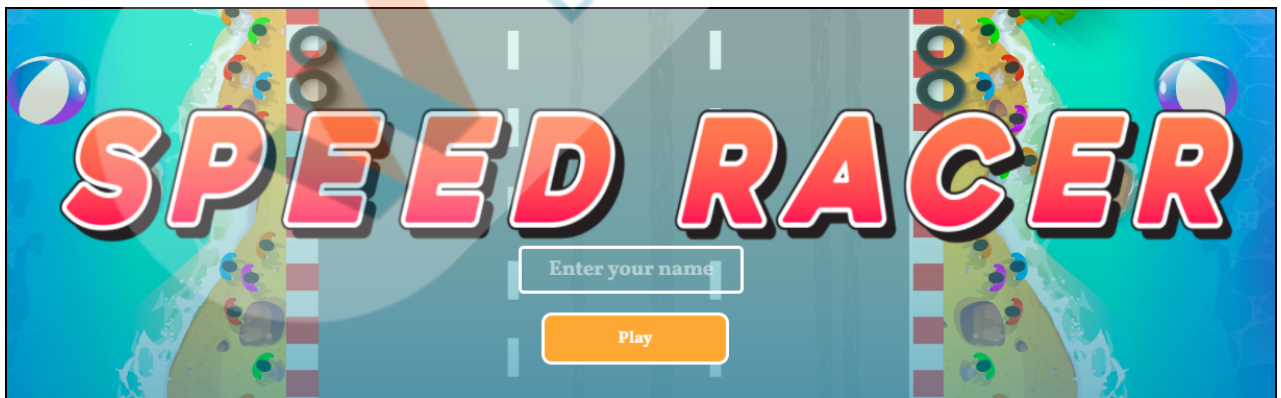
```

16 .gameTitle{
17   width: 85%;
18   height: 160px;
19 }
20
21 .greeting{
22   font-size: 40px;
23   color: white;
24   font-family: vollkorn;
25   text-align: center;
26 }
27
28 .customInput {
29   width: 200px;
30   height: 30px;
31   border: 3px solid white;
32   border-radius: 5px;
33   padding: 5px;
34   background-color: transparent;
35   font-size: 20px;
36   outline: none;
37   color: white;
38   font-family: vollkorn;
39   text-align: center;
40 }
41
42 ::placeholder /* Chrome, Firefox, Opera, Safari
43   color: white;
  
```

[Teacher also explains the function `hide()` from `form.js`]
`.hide()` is used with each element to hide/ remove the element from the canvas.



The student observes and asks questions.

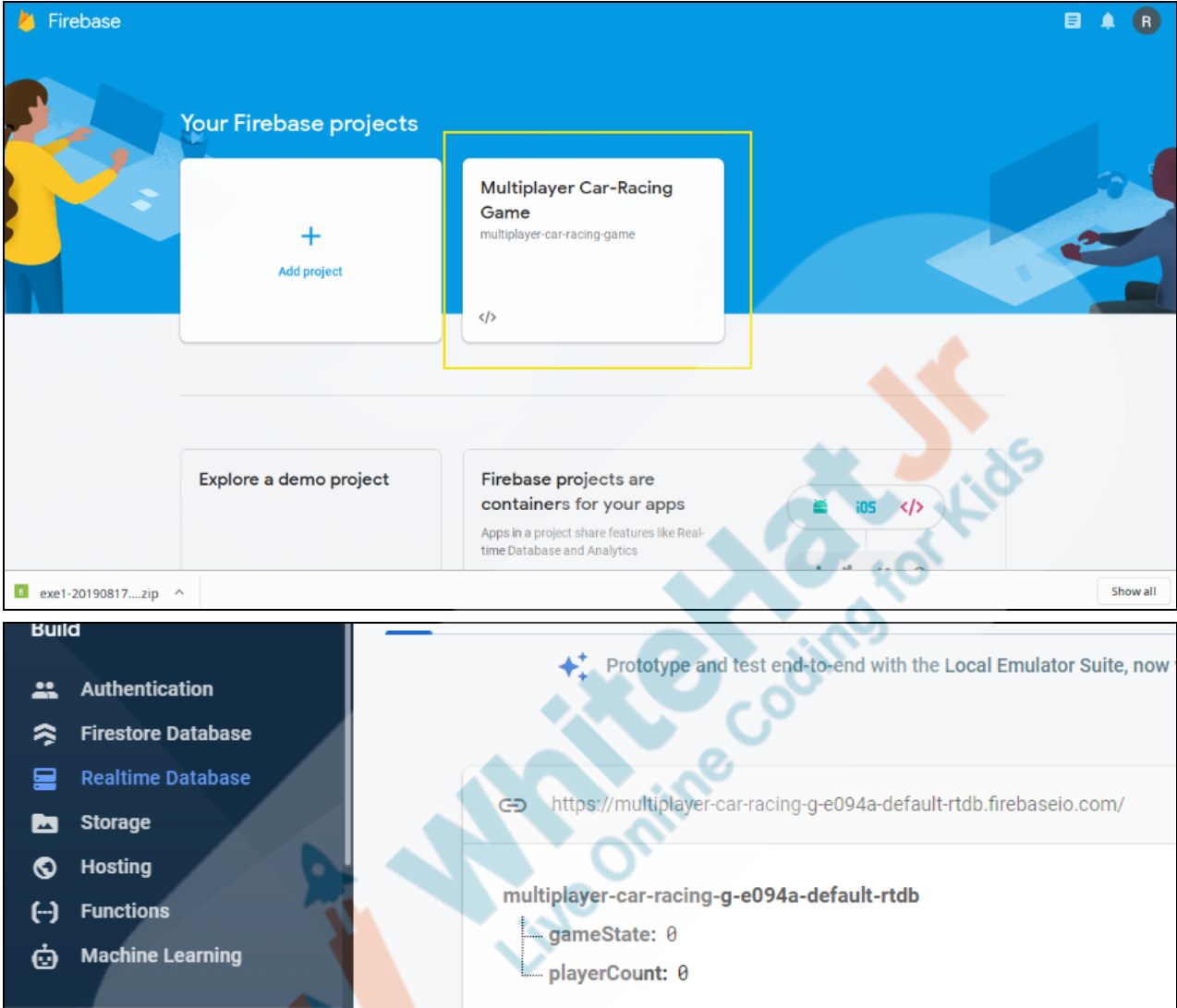
[The teacher clicks on GoLive and shows the output.]



Teacher Stops Screen Share

STUDENT-LED ACTIVITY - 20 mins

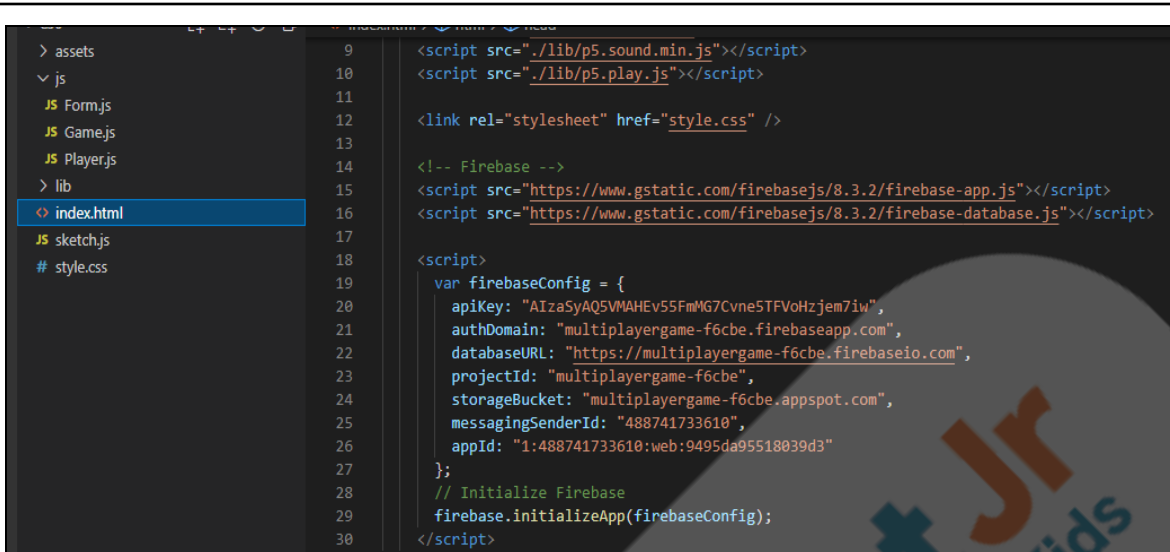
<ul style="list-style-type: none"> • Ask Student to press the ESC key to come back to the panel • Guide the student to Start Screen Share • The teacher gets into Fullscreen 	
<p style="text-align: center;">ACTIVITY</p> <ul style="list-style-type: none"> • Create a database and configure it with an index.html • Create an action on form button 	
<p style="text-align: center;">Teacher starts slideshow  from slide 20 to slide 24.</p>	
Run the presentation for slide 20 to slide 24 to set the student activity context.	
<p style="text-align: center;">Teacher ends slideshow </p>	
Teacher Action	Student Action
<p><i>Guide the student to open the Student Activity 1 and download the zip folder, unzip it and save as C36.</i></p> <p><i>During the class, if a student is changing any position of any element or object in the code/ ask them to write it somewhere to use in the codes of the next classes.</i></p>	<p><i>Alternatively, the student can clone Student Activity 1.</i></p>
<p>Let's first start with modifying the database.</p> <p><i>Guide the student to login to console.firebase.google.com and modify the previous database /or to create a new database structure equivalent to the following:</i></p>	<p><i>The student logs in to the firebase console and creates the database structure.</i></p>



The screenshot shows the Firebase console interface. At the top, the 'Your Firebase projects' section displays a project named 'Multiplayer Car-Racing Game' with the ID 'multiplayer-car-racing-game'. Below this, the 'Explore a demo project' section shows the Firebase projects are containers for your apps. The 'Build' section on the left lists various services: Authentication, Firestore Database, Realtime Database, Storage, Hosting, Functions, and Machine Learning. The main content area shows the 'Prototype and test end-to-end with the Local Emulator Suite, now' section, which includes a link to the Realtime Database: <https://multiplayer-car-racing-g-e094a-default-rtdb.firebaseio.com/>. Below the link, the database structure is shown as a JSON object:

```
{  "gameState": 0,  "playerCount": 0}
```

The image and code given below are for reference only; use SDK from the database created by the teacher/student.



```

9      <script src="./lib/p5.sound.min.js"></script>
10     <script src="./lib/p5.play.js"></script>
11
12     <link rel="stylesheet" href="style.css" />
13
14     <!-- Firebase -->
15     <script src="https://www.gstatic.com/firebasejs/8.3.2/firebase-app.js"></script>
16     <script src="https://www.gstatic.com/firebasejs/8.3.2/firebase-database.js"></script>
17
18     <script>
19       var firebaseConfig = {
20         apiKey: "AIzaSyAQ5VMAHEv55FmMG7CvneSTFVoHzjem7iw",
21         authDomain: "multiplayergame-f6cbe.firebaseio.com",
22         databaseURL: "https://multiplayergame-f6cbe.firebaseio.com",
23         projectId: "multiplayergame-f6cbe",
24         storageBucket: "multiplayergame-f6cbe.appspot.com",
25         messagingSenderId: "488741733610",
26         appId: "1:488741733610:web:9495da95518039d3"
27       };
28       // Initialize Firebase
29       firebase.initializeApp(firebaseConfig);
30     </script>
  
```

Generate the SDK code and copy the same in **index.html**.
[The steps are covered in detail in C35]

Ask the student to see the output before moving on; s/he should be able to see the input box and click on the button.

The student generates SDK code and pastes it in index.html.

The student clicks on goLive to see the output and clicks on the button but nothing happens.

What should happen when the button is clicked?
[Probe student to give answer]

For doing so, we will have to write database-related functions in **player.js**. We shall do that in the next class, for today, we will learn how to take action when the button is pressed by the mouse.

ESR:

The player name should get saved in the database & the player count should increase.

Players will press the button using the mouse; in p5 there is a **mousePressed()** function which is called whenever the mouse is pressed.

*Guide the student to open Student activity 3 to refer to **mousePressed()**.*

Before using that, let us understand one more concept of Arrow function.

At this point the teacher can either share her screen and use the P5 editor to explain the arrow function or the teacher can write the same on VSC in case of live sharing; One example is given below to explain the syntax of an arrow function.

Student clicks
[\[student activity 3\]](#)

*** For teacher reference NOT PART OF CODE example of Arrow function**

```
//Currently we are using this.
display(){

  this.button()
}
function mousePressed(){
  //Action to be taken when mouse is pressed
  //This function will be activated wherever mouse is pressed
  //on screen. To bind it with that particular button we use arrow
  function
}

//function word gets replaced by => Arrow
//This is syntax for arrow function
//This way function is bound with particular button only;
display(){
  this.button.mousePressed(()=>{

  })
}
```


When the button is pressed, we want to greet the player when the player writes their name and logs in.

We also want to update the playerCount and the player name in the database, this part we will cover in the next class.

button.mousePressed() can be used to trigger an action when a mouse is pressed on the button. It expects a function as an argument.

Let's write the code to display a greeting and update the database when the button is pressed.

We will create a new function **handleMousePressed(){ }** Remember to call this function in **display()**.

*The student writes the **button.mousePressed()** function.*

When the button is pressed, the student writes code to:

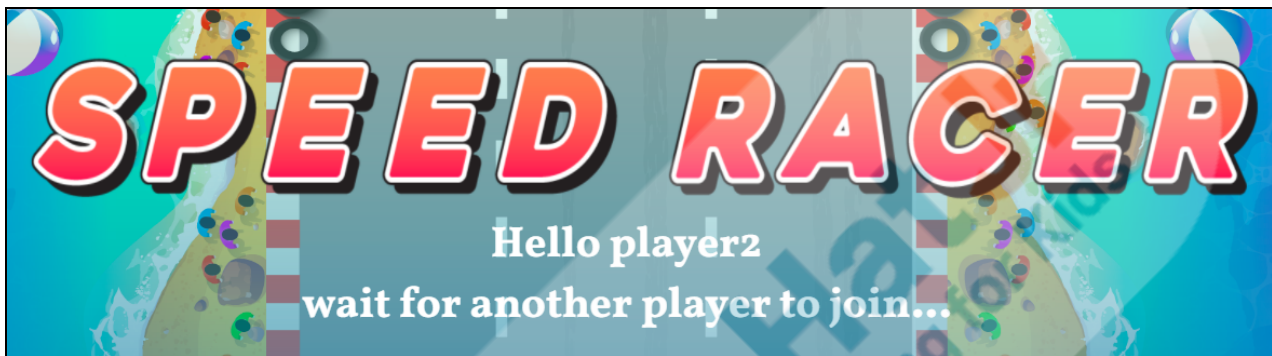
- Hide the input and the buttons.*
- Display the greetings element and use it to greet the player when the player has logged in.*

```
handleMousePressed() {  
  this.playButton.mousePressed(() => {  
    this.input.hide();  
    this.playButton.hide();  
    var message = `  
    Hello ${this.input.value()}  
    </br>wait for another player to join...`;   
    this.greeting.html(message);  
  });  
}
```

```
display() {  
  this.setElementsPosition();  
  this.setElementsStyle();  
  this.handleMousePressed();  
}
```

Let's test our code.

The student runs the code using GoLive- The student opens the link in 2 different browsers.



Help the student debug any errors.

What do you see as the problem here?

Why do you think the players are not added to the database?


Allow students to go through the code to find the reason.




Correct, additionally we will also have to write different functions to add a player count and player information in the database.

As a challenge, why don't you think of a solution for these methods before our next class?

ESR: We added the name of the player but it is not reflected in the database.

ESR:
We have not given .set() to add player information in the database.

Teacher Guides Student to Stop Screen Share	
WRAP-UP SESSION - 5 Mins	
 <p>Teacher starts slideshow from slides 25 to slide 35. Refer to speaker notes and follow the instructions on each slide.</p>	
Activity details	Solution/Guidelines
<p>Run the presentation from slide 25 to slide 35</p> <p>Following are the WRAP-UP session deliverables:</p> <ul style="list-style-type: none"> ● Revise the concepts ● WRAP-UP Quiz ● Explain the facts and trivia ● Project for the day ● Next class challenge 	<p><i>Guide the student to develop the project and share it with us.</i></p>
Quiz time - Click on the in-class quiz	
Question	Answer
<p>Which object holds the state of the game (0:WAIT; 1:PLAY; 2:END)?</p> <ul style="list-style-type: none"> A. Form Object B. Player Object C. Game Object D. None of these 	<p>C</p>
<p>Which language is used to create the content like a form on a page?</p> <ul style="list-style-type: none"> A. JavaScript B. HTML C. CSS D. JAVA 	<p>B</p>

<p>Which object should be created every time a new player logs in?</p> <p>A. Form Object</p> <p>B. Player Object</p> <p>C. Game Object</p> <p>D. None of these</p>	<p>B</p>
<p>End the quiz panel</p>	
<p><u>FEEDBACK</u></p> <ul style="list-style-type: none"> • Appreciate the student for their efforts in the class. • Ask the student to make notes for the reflection journal along with the code they wrote in today's class. 	
TEACHER ACTION	STUDENT ACTION
<p>You get Hats off for your excellent work!</p> <p>We have just created a form to register our players and their names in the database.</p> <p>We need to do a number of things more - we need to stop the player addition after 2 players, we need to change the game state to play, we need to create a car racing game between 2 players. We will be writing more code for all this in the coming classes.</p>	<p><i>Make sure you have given at least 2 Hats Off during the class for:</i></p> <div data-bbox="1019 1115 1312 1213"> <p>Creatively Solved Activities  +10</p> </div> <div data-bbox="1019 1234 1312 1333"> <p>Great Question  +10</p> </div> <div data-bbox="1019 1354 1312 1453"> <p>Strong Concentration  +10</p> </div>

*** This Project will take only 30 mins to complete.
 Motivate students to try and finish it immediately after the class.***

Project Name:

MYQUIZ FORM

Goal of the Project:

In Class 36, you used p5.DOM.elements to create an HTML form. In this project, you will use the concepts for creating elements using p5.DOM in today's project to create a form for the MyQuiz game.

Story:

Prakriti loves asking quizzes and she always tries to find unique questions and ask different people. But now, she is thinking of creating her own multiplayer quiz game where she can ask a quiz question to different people at the same time.

Can you help her in creating the game?

I am very excited to see your project solution and I know you will do really well.

Bye Bye!

Note: You can assign the project to the student in class itself by clicking on the Assign Project button which is available under the projects tab.

The student engages with the teacher over the project.

Teacher ends slideshow



Teacher Clicks

✕ End Class

ADDITIONAL ACTIVITIES

Additional Activities

Encourage the student to write reflection notes in their reflection journal using Markdown.

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?

The student uses the Markdown editor to write their reflections in a reflection journal.

Activity	Activity Name	Links
Teacher Activity 1	p5 BoilerPlate for Multiplayer Car Racing Game	https://github.com/pro-whitehatjr/C36RV_SpeedRacer_TeacherActivity/
Teacher Activity 2	p5 dom Reference	https://p5js.org/reference/#group-DOM
Teacher Activity 3	Teacher's Reference code	https://github.com/pro-whitehatjr/C36RV_SpeedRacer_Reference_Code
Teacher Activity 4	MultiPlayerCar Racing Game	https://pro-whitehatjr.github.io/C42-SpeedRacer_Reference-Code/
Student Activity 1	p5 BoilerPlate for Multiplayer Car Racing Game	https://github.com/pro-whitehatjr/C36RV_SpeedRacer_StudentActivity
Student Activity 2	p5 dom reference	https://p5js.org/reference/#group-DOM
Student Activity 3	MousePressed()	https://p5js.org/reference/#/p5/mousePressed

Teacher Reference	Visual aid link	https://curriculum.whitehatjr.com/Visual+Project+Asset/PRO_VD/PRO_C36_V3_lit_With Cues.html
Teacher Reference	In-class quiz	https://s3-whjr-curriculum-uploads.whjr.online/09af4b42-3809-45eb-8468-32038abf544f.pdf
Project Solution	MyQuiz Form	https://github.com/pro-whitehatjr/PRO-C36_ProjectSolution