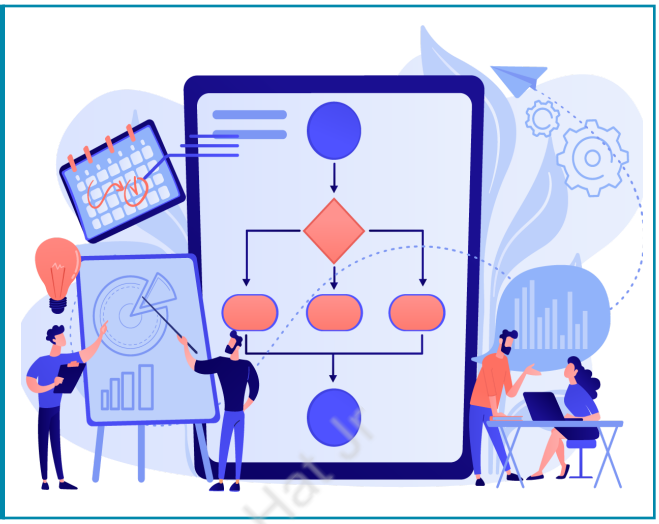
STRUCTURING BEFORE CODING



What is our GOAL for this MODULE?

In this class, we learned to design a form using p5 dom to allow players to log in. We used the OOPs programming style to write the code.

What did we ACHIEVE in the class TODAY?

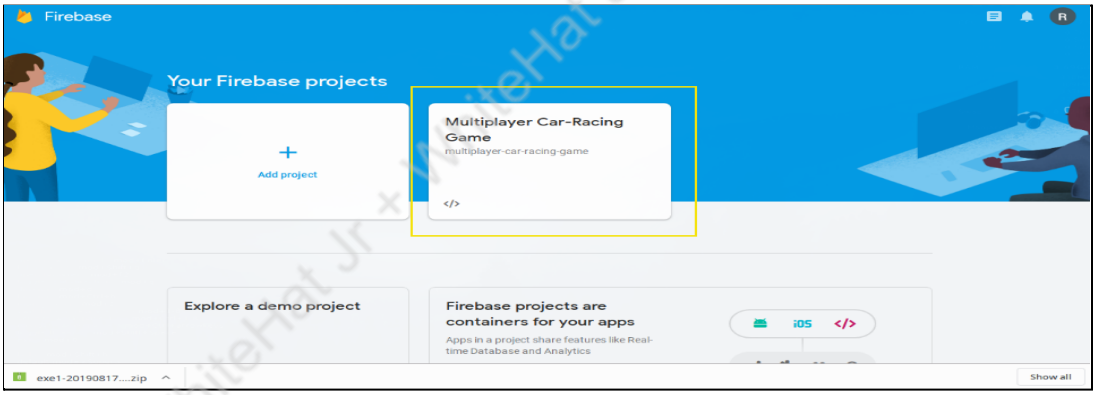
- Created structure of Code and Database
- Created Form using p5.DOM.Elements. (HTML in Javascript)
- Created a canvas size that is responsive to the device size.

Which CONCEPTS/ CODING BLOCKS did we cover today?

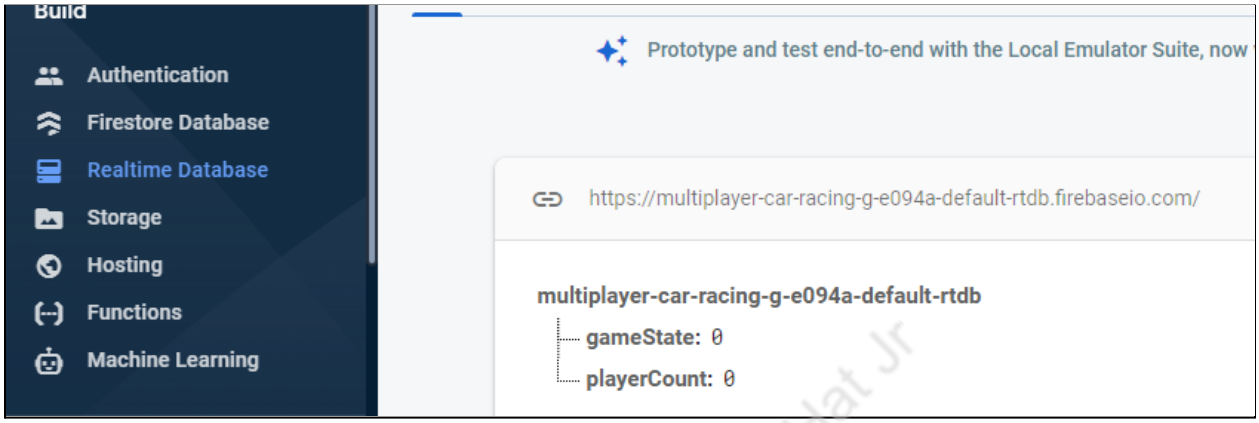
- About p5.DOM.elements
- Creating a Form
- Creating database fields
- OOPs programming concept
- mousePressed() & arrow function

How did we DO the activities?

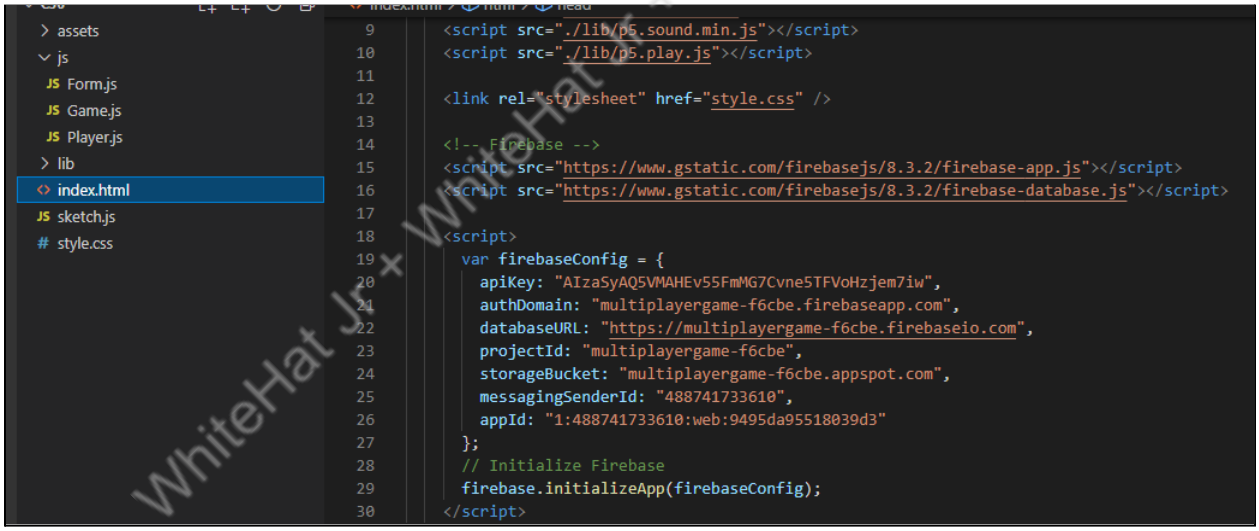
1. Created three class for making a multiplayer game:
    - Form: The form contains the input box, a button to log in. When the button is pressed, the player's name gets registered in the database and a new player is created.
    - Player: A new player object is created every time a new user logs in. It should contain all the information about the player - name, position in the game, and so on. For now, it can just have the name property. It should also be able to read and write player information to the database - for example player count or player name.
    - Game Object: The game object should be able to hold the state of the game. It should be able to display the form when the game state is 0 (WAIT), the game when the game state is 1 (PLAY), leaderboard when the game state is 2 (END). For now, we will only consider the case when the game state is 0.
2. Create a firebase database:
    - Open the firebase console.
    - Start a new project.



- Create a new database and nodes for **gameState** and **playerCount** with an initial value of 0.



- Generate SDK code and add the database SDK (Software Development Kit) to **index.html**.



3. Declare global variables for class Game.
    - Initialize database.
    - Create a canvas with **windowWidth, windowHeight.**
    - Preload background image.
    - Create a game object.
    - Create a **windowResized()** function to make screen size responsive to device size.

```
var canvas;
var backgroundImage;
var bgImg;
var database;
var form, player;
var playerCount;

function preload() {
  backgroundImage = loadImage("./assets/background.png");
}

function setup() {
  canvas = createCanvas(windowWidth, windowHeight);
  database = firebase.database();
  game = new Game();
  game.start();

}

function draw() {
  background(backgroundImage);
}

function windowResized() {
  resizeCanvas(windowWidth, windowHeight);
}
```

4. The game object was created in **setup()** and called the **start()** function.
    - **start()** function which starts the game is displayed on the screen depending on the state of the game.
    - When the game state is 0, we wanted a form and a player object to be created.
    - Display the form and get the player's name.
    - Write code to create these objects even though the blueprint was not defined yet. This is called writing code using abstraction.

```
class Game {
  constructor() {}

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

5. HTML was 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 Document Object Model (or DOM). We used the p5 Dom library to create the form.

```
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");
}
```

- Display each element with position and style them.

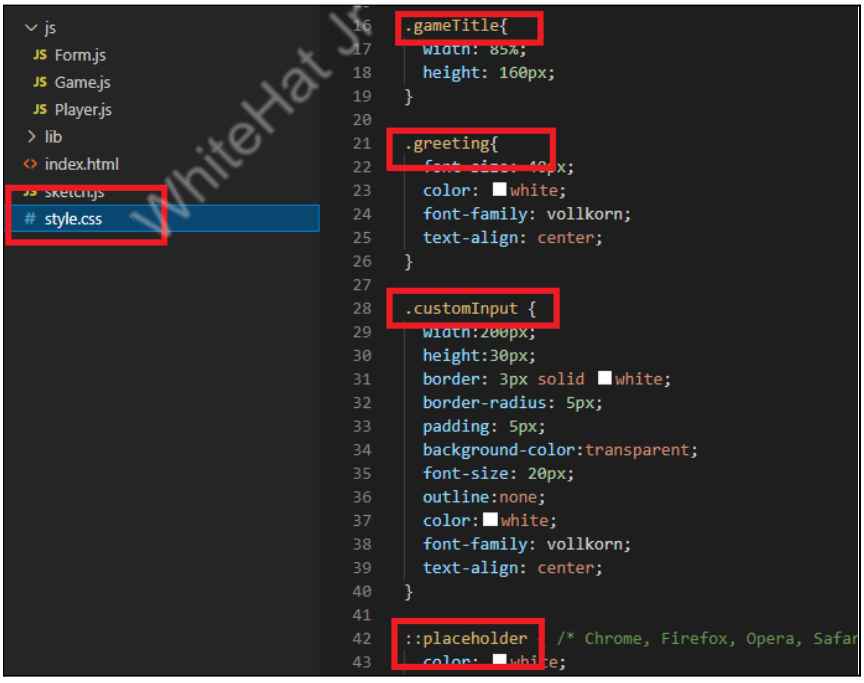
```
setElementsPosition() {
  this.titleImg.position(120, 160);
  this.input.position(width / 2 - 110, height / 2 - 80);
  this.playButton.position(width / 2 - 90, height / 2 - 20);
  this.greeting.position(width / 2 - 300, height / 2 - 100);
}

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

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

}
```

6. Function **setElementsStyle()** uses the styles defined in **style.css**.



7. **button.mousePressed()** was used to trigger an action when a mouse button was pressed. It expects a function as an argument.

```
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);
  });
}
```

8. Call the **handleMousePresssed()** function in the **display()** method.

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

9. Run the code to check for bugs and debug them.

OUTPUT

What's next?

In the next class, you will be creating more database queries, to add both players' details. Read & write game state and player count.

EXTEND YOUR KNOWLEDGE:

Watch this video to learn more about creating forms using p5.DOM.js:
https://youtu.be/lAtoaRz78I4