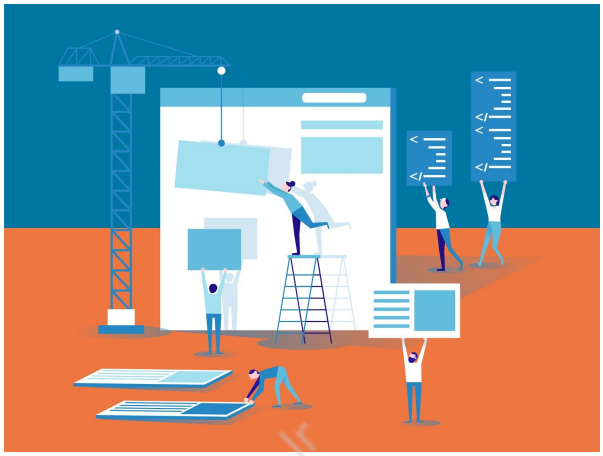
PRO-C36

WhiteHat Jr
Live Online Coding for Kids

Structuring before coding



What is our GOAL for this MODULE?

The goal for this module is to explore the OOPs programming style and create form using p5 dom and log the players in the database.

What did we ACHIEVE in the class TODAY?

- We designed a form using p5 dom to allow players to login and log the player names to the database.
- The gamestate and the playercount are also logged.
- We used the OOPs programming style to write the code.

Which CONCEPTS/CODING BLOCKS did we cover today?

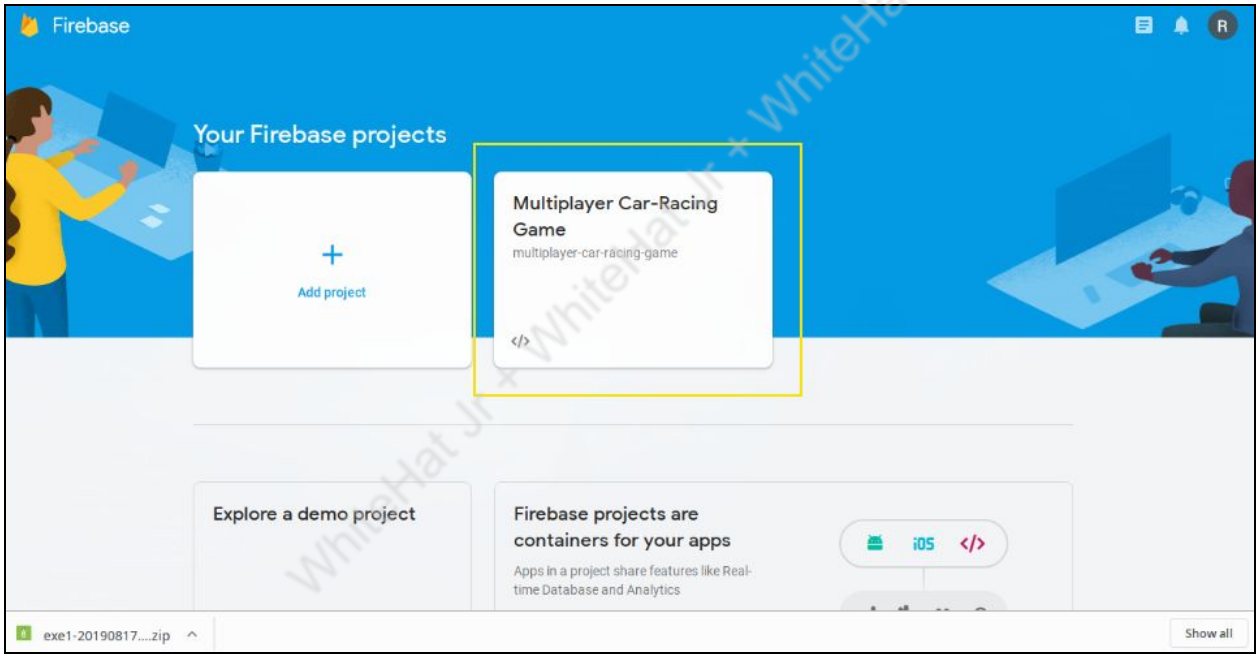
- OOPs programming concept
- Firebase database
- Game states
- P5 dom

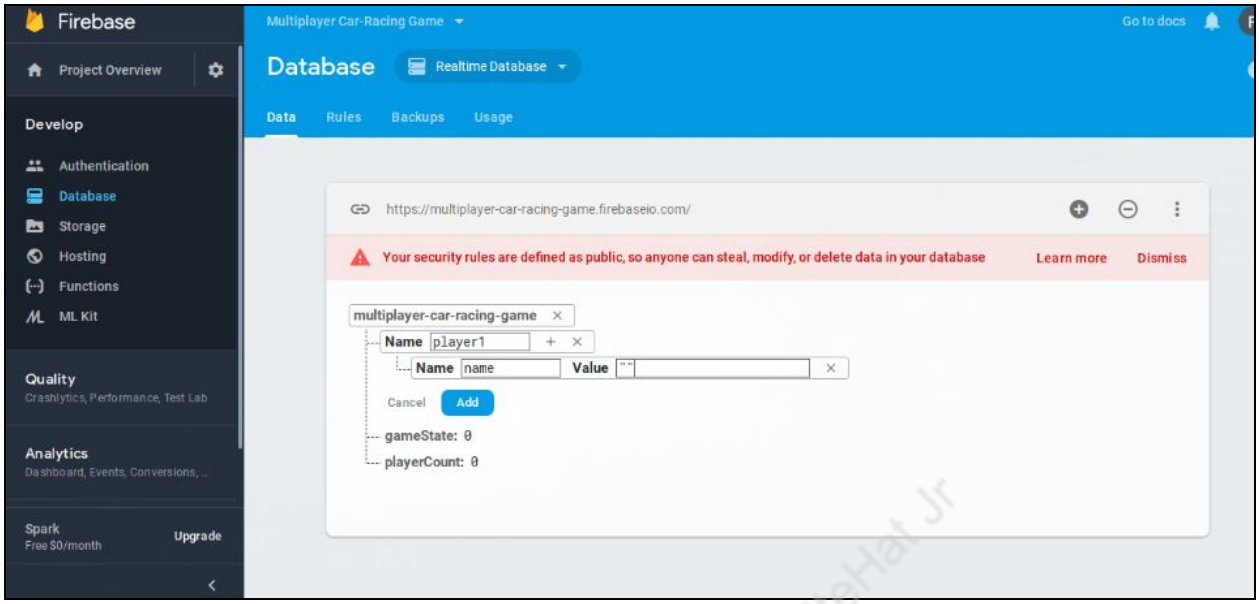
How did we DO the activities?

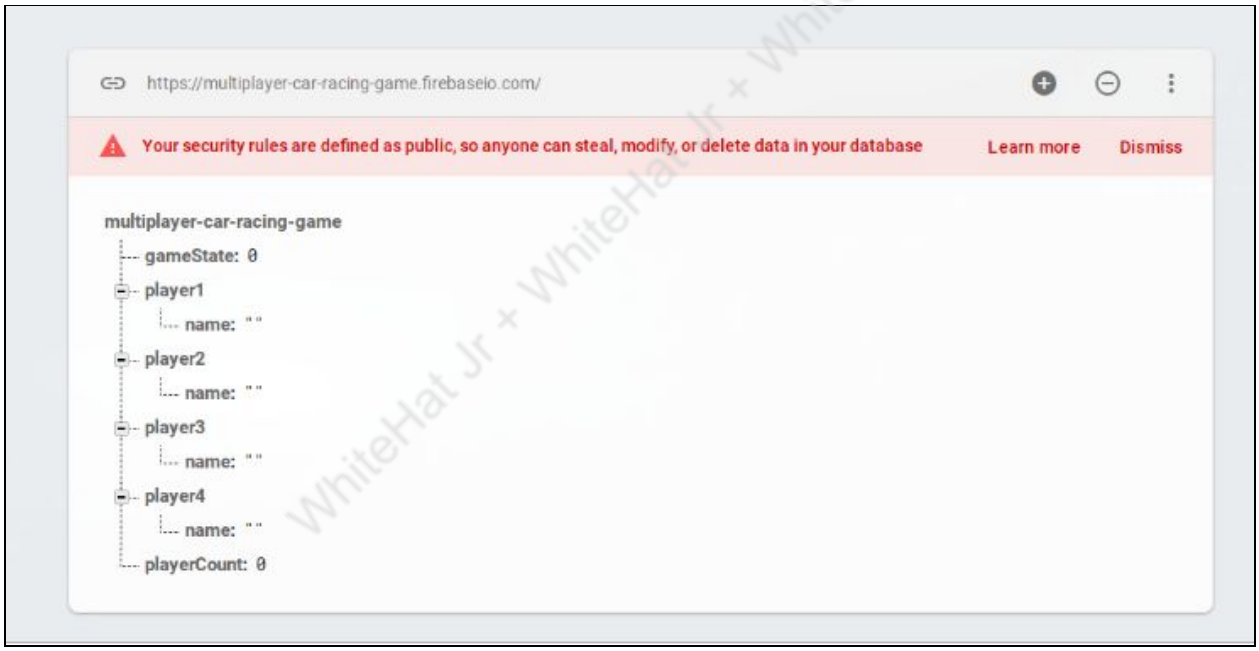
Used p5 dom and created a login form for players to log in.

We needed to have at least 3 objects:

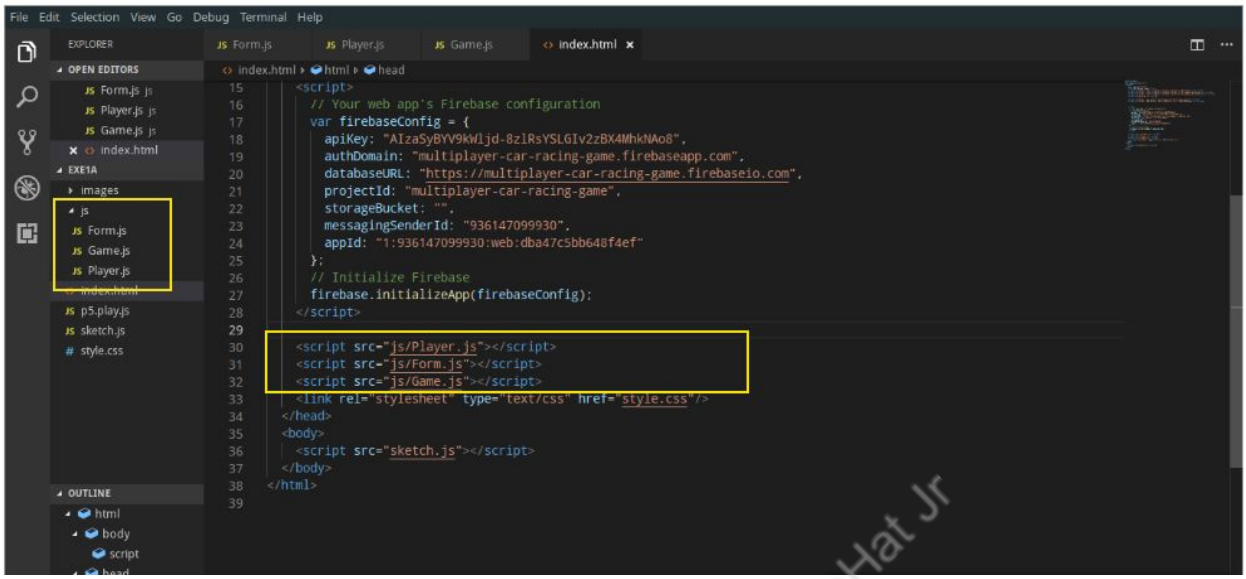
1. Form: Form should contain the input box, a button to log in. When the button is pressed, the player's name should be registered in the database and a new player should be created.
2. Player: A new player object should be created every time a new user logs in. It should contain all the information about the player - name, position in the game etc. 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.
3. Game Object: 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) or the game when the game state is 1 (PLAY) or leaderboard when the game state is 2 (END). For now, we will only consider the case when the game state is 0.



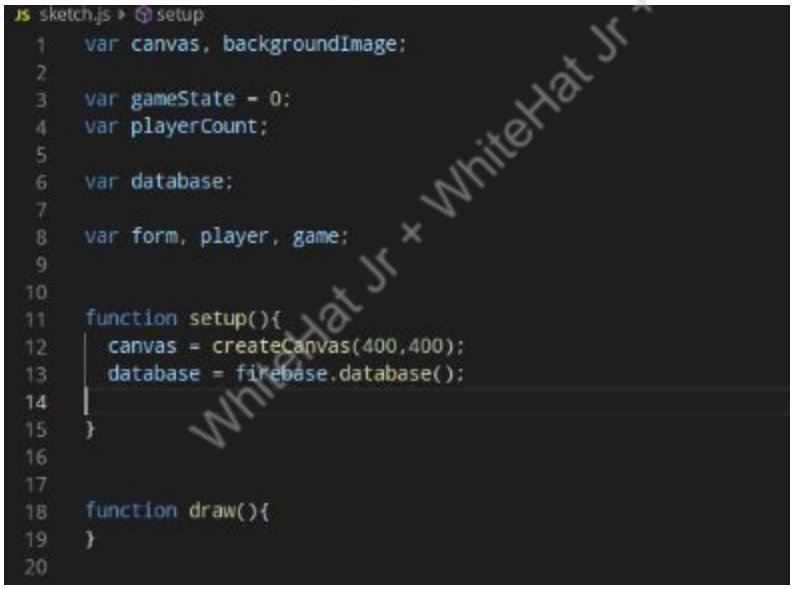




We created a new folder in our directory called js which contained the blueprint of all the 3 objects in our game - Game, Form and Player.



The sketch.js file that included all the global variables.



Game object was able to read the gameState and update the gameState. It also started itself and displayed the game on the screen depending on the gameState.

Constructor of a class was used to give properties to an object when it was created. We kept the constructor empty.

Then we wrote functions inside the Game Class to getState and updated the state.

- getState() simply read the game state from the database.

- update(state) updated the gameState in the database to a value passed to it inside the parentheses.

-> databaseReference.on() creates a listener which keeps listening to the gameState from the database. When the gameState is changed in the database, the function passed as an argument to it is executed.
Note: Here the function is directly written inside the .on() listener.
-> databaseReference.update() will update the database reference. Here "/" refers to the main database inside which gameState is created.

Also, we created a start() function which started the game and 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. We wanted to display the form and get the playerCount. We wrote code to create these objects even though the blueprint was not defined yet. This is called writing code using abstraction.

```
js ▸ JS Game.js ▸ Game ▸ getState
1   class Game {
2     constructor(){}
3
4     getState(){
5       var gameStateRef  = database.ref('gameState');
6       gameStateRef.on("value",function(data){
7           gameState = data.val();
8       })
9
10    }
11
12    update(state){
13      database.ref('/').update({
14        gameState: state
15      });
16    }
17
18    start(){
19      if(gameState === 0){
20        player = new Player();
21        player.getCount();
22        form = new Form()
23        form.display();
24      }
25    }
26  }
27
```

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 is 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.

We wanted to greet the player, when the player writes their name and logs in. We also wanted to update the playerCount and the player name in the database.

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

We wrote the code for the Player Class.
We wrote a function getCount() to get the playerCount and updateCount() to update the playerCount in the database.
To update the player name in the database: We needed to create new entries in the database. We did this using string concatenation. If the playerCount was 1, we created a database entry for player1 and we set the name for it and so on

Additional code in our sketch.js file created a new Game object, got the gameState and then started the game.

We ran the code and checked for bugs and debugged them.

What's next?

In the next class, you will be creating a multiplayer car racing game.

EXTEND YOUR KNOWLEDGE:

Watch this video to learn more about creating forms in p5:
https://youtu.be/lAtoqRz78I4