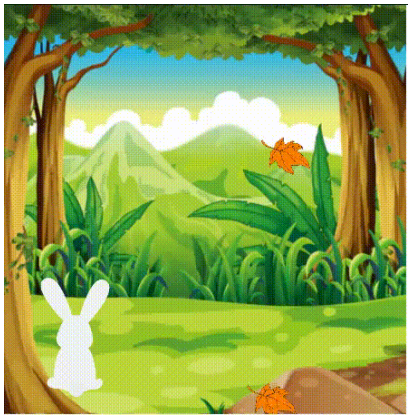
Topic	MEMORY LEAK AND SWITCH STATEMENT	
Class Description	**The student learns about memory leak and how it can be avoided in a well written program. The student uses switch statements in the code to randomly spawn different varieties of obstacles and also build a simple scoring system.**	
Class	**C13**	
Class time	**45 mins**	
Goal	• Correct the memory leak problem in code. • Use switch case statements to randomly spawn different kinds of obstacles in the game. • Design a simple scoring system.	
Resources Required	• Teacher Resources: ○ VS Code Editor ○ Laptop with internet connectivity ○ Earphones with mic ○ Notebook and pen • Student Resources: ○ 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**	**10 mins** **10 mins** **20 mins** **5 mins**
WARM-UP SESSION - 10mins		
Teacher starts slideshow  **from slides 1 to 17**		
Activity details	**Solution/Guidelines**	

Hey <student name>. How are you? It's great to see you! Are you excited to learn something new today? **Run the presentation from slide 1 to slide 4.** **Following are the warm up session deliverables** ● Connect students to the previous class. ● Talk about Memory Leak	**ESR**: Hi, thanks, yes I am excited about it! Click on the slide show tab and present the slides.
Q&A Session	
Question	**Answer**
Select the correct block of code to generate a random number between 1 to 3. A. `var select_sprites = Math(random(1,3));` B. `var select_sprites = Math.random(random(1,3));` C. `var select_sprites = Math.round(1,3);` D. `var select_sprites = Math.round(random(1,3));`	D

What will the following code block do?
`if (frameCount % 80 == 0) {`
`  if (select_sprites == 1) {`
`  createApples();`
`  } else if (select_sprites == 2) {`
`  createOrange();`
`  }`
`}`
	A



A. It will call the **createApples( )** and **createOrange( )** functions after every 80 frames.
B. It will call the **createApples( )** and **createOrange( )** functions after every 50 frames.
C. It will call the **createOrange( )** function only after every 80 frames.
D. It will call the **createApples( )** function only after every 80 frames.

Continue the warm-up session

Activity details	Solution/Guidelines
Run the presentation from slide 5 to slide 17 to set the problem statement. • Introduce memory leak and how it can be avoided in a well written program. • Spawn different kinds of obstacles on the way in the Trex runner game. • Steps to write and run the code.	Narrate the story by using hand gestures and voice modulation methods to bring in more interest in the student.

Teacher ends slideshow

TEACHER-LED ACTIVITY - 10 mins

ACTIVITY

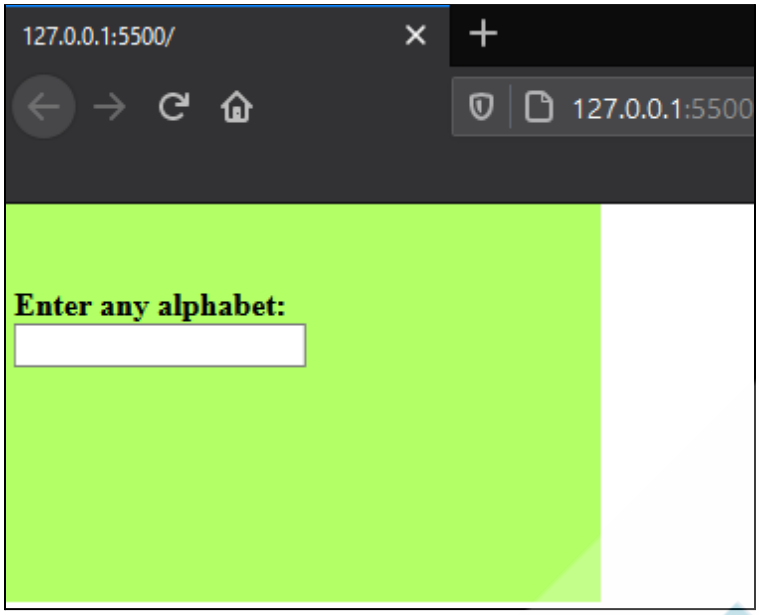
- **Learn about switch case statements**
- **Set lifetime for game objects.**

Teacher Initiates Screen Share	
Teacher Action	**Student Action**
Step- 1 **Teacher-led Activity-1** In previous classes we have learned about if and **if-else** conditions. These conditions are used to change the flow of the code such as, based on one condition, the program will behave in one way and another way when different conditions come. The **if-else** is not the only way to do so. We have one more method that can perform the same thing and make our code more readable. It is called the switch case statement. So how does it work? Any guesses?	*Download the blank p5.js project from Template Code* **ESR:** Varied.
It's actually pretty simple. In our home we have multiple switches for different appliances and when you press one switch a specific equipment runs. In the same way, in switch case statements we have a switching statement which is input value. Now, that input value can have many cases or types. Based on the case/ type of switching statement we execute a piece of the code.	
You will have more clarity on this as we move forward with the activity.	

The teacher writes the code.	

```
var input, heading;

function setup() {

  createCanvas(300, 200);
  background(178,255,102);

  input = createInput();
  input.position(5, 60);

  heading = createElement('h4', 'Enter any alphabet:');
  heading.position(5, 20);

  textAlign(CENTER);
  textSize(50);
}
```

First we will create an input dialog box to get the user input. We declare two variables—input and heading. Then using the **createInput()** function we will create a text input box on the canvas. Then we will specify the position of the input box using the **input.position()** function. We will create a text heading for the text input box using **createElement()** function. In the **createElement()** function we need to specify the heading size such as 'h4' and the text for the heading. Using the position function we position the heading on the canvas. We will also set the size and the alignment of the text. *Teacher runs the code to show the output.*	Student observes and learn
Output:	



Users can enter the value here; to read the value entered by the user, we will use **input.value()** function and store this in a variable.

Now we will use our switch statement. We are getting a character from the user, now we need to tell whether this character is a vowel or consonant.

We will define the **switch(value)** which means we'll be switching the response based on the value. Then we need to define the possible case and response.

```
function draw() {

  const value = input.value();
  switch (value) {

    case 'a':
      console.log("Vowel");
      break;

    case 'e':
      console.log("Vowel");
      break;

    case 'i':
      console.log("Vowel");
      break;

    case 'o':
      console.log("Vowel");
      break;

    case 'u':
      console.log("Vowel");
      break;

    default:
      console.log("Please enter any char
acter");
  }
}
```

Which is defined with **case** keyword In our case we are looking for the vowels. Which characters are the vowels?	**ESR:** a, e, i, o, u

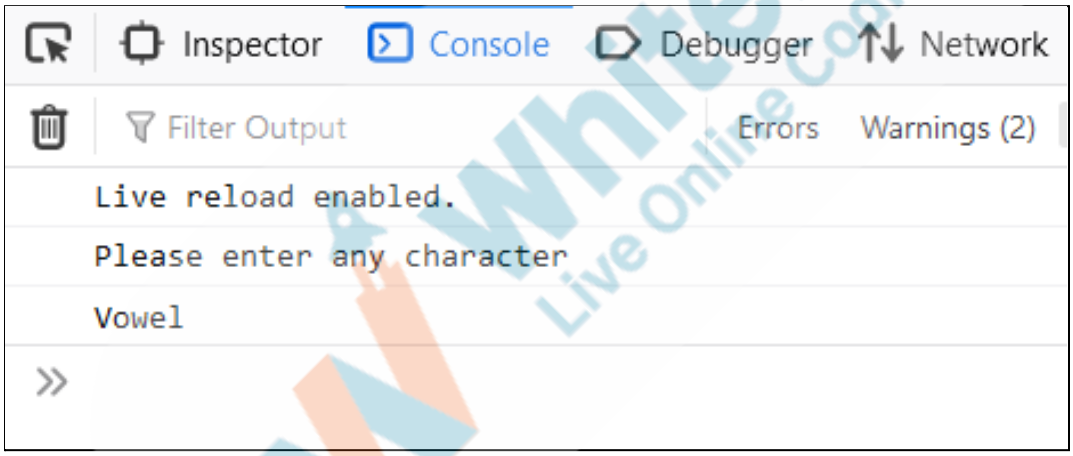
Yes, great! So in the parentheses of the switch statement we will write our cases. The case will be like a condition with which we will compare the user input. The first case is ‘a’ and pay attention to the syntax we are writing the keyword case then space and then the ‘a’ is written in the quotes and after that we will add a colon. This tells us that the case statement is complete now. In the next line, we will write what will happen if our input matches the case value i.e the user enters the letter ‘a’. Here we will say **Console.log(“Vowel”)** because it is a vowel; then the next statement is very important, that is **break.** Break statement is used to terminate a running code statement. Once we compare the user input with our case ‘a’ and both of them match, our code comes to a break statement, our switch statement sequence will be terminated. If a user has entered any other character except “a” then our code will move to the next case. *The teacher writes the code for the next 4 cases.*	
Now we have written the code for all the cases. Is our code complete? Is there anything remaining?	*Student thinks about the answer.*
We have written the case for vowels, but what will happen if the user enters any other character which is not a vowel? We have to add that condition also in our code.	

To solve this issue we have a default statement which gives a default response when the user input does not match with any of the cases.

To write a default statement we simply write the default keyword and then colon. In the next line will write what happens when the user enters any other letter instead of the vowel; here we are saying any character.

Because we are writing this in a **draw()** function, and this function is continuously running so we have to prompt the user to enter a character.

Note: Please enter a single character in the input box at one time. In case you want to add a new character please delete the first one.



We have seen how to use the switch case statement to check whether a character is a vowel or not.

We can use the switch case for a wide variety of applications. We are going to use a switch case to show the obstacles in our T-rex game.

We are creating sprites inside the spawnClouds function. And they live even after they have gone outside the screen. These sprites occupy space in the computer's memory. We can follow two approaches here ● We can find out when the clouds go across the screen and then destroy them to free space in the memory. OR ● Assign lifetime to the variables so that they destroy themselves after their lifetime is over. The second approach would be useful and easier here. Let us assign some random lifetime (say 50) to the cloud variable and see what happens. **Note:** 50 is a random number here. We can do this by calling **cloud.lifetime = 50.** *Teacher writes and runs the code.*	*Download the code from Teacher Activity 2* **ESR:** Student observes.

```
function spawnClouds() {
  //write code here to spawn the clouds
   if (frameCount % 60 === 0) {
     cloud = createSprite(400,320,40,10);
     cloud.addImage(cloudImage);
     cloud.y = Math.round(random(10,60));
     cloud.scale = 0.4;
     cloud.velocityX = -3;

     //assign lifetime to the variable
     cloud.lifetime = 50;

     //adjust the depth
     cloud.depth = trex.depth;
     trex.depth = trex.depth + 1;
    }
}
```

	What do you see happening?	**ESR:** The clouds are disappearing midway.
	After every frame, the **draw()** function is called and the lifetime of the cloud reduces by **1**. The cloud vanishes when the lifetime becomes **0**. The memory space in the computer also gets freed up.	*The student asks questions to clarify.*
	Clouds have to cover the width of **600** pixels at a velocity of **3**. What should be the lifetime of the clouds?	**ESR:** Time = Distance/Speed 600/3 = 200 secs
	Let us assign this lifetime to each cloud variable which is getting created. *Teacher writes and runs the code.*	*Student observes the output.*

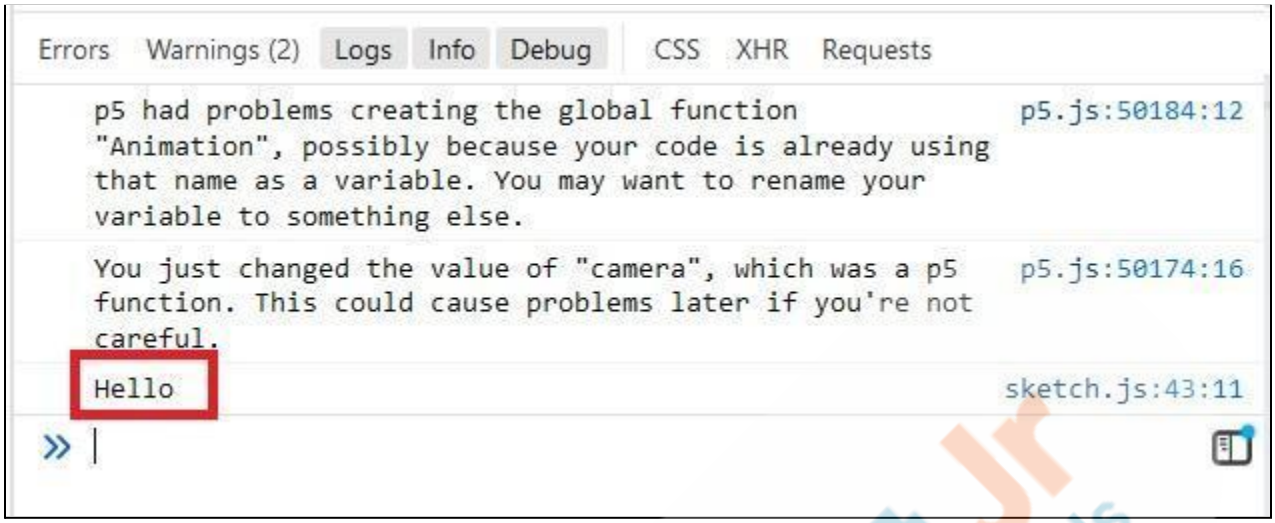
```
function spawnClouds() {
  //write code here to spawn the clouds
  if (frameCount % 60 === 0) {
    cloud = createSprite(600,100,40,10);
    cloud.y = Math.round(random(10,60));
    cloud.addImage(cloudImage);
    cloud.scale = 0.5;
    cloud.velocityX = -3;

    //assign lifetime to the variable
    cloud.lifetime = 200;

    //adjust the depth
    cloud.depth = trex.depth;
    trex.depth = trex.depth + 1;
  }
}
```

What do you see happening?	**ESR:** The cloud floats through the width and then vanishes near the edge.
This means that the space in the computer memory is continuously getting freed up and there is no memory leakage.	*Student listens.*
We would like to learn one more important aspect and then you can generate/spawn the obstacles for Trex just like we generated them for the clouds.	*Student listens.*
When we store any text information in a computer, we write it inside quotes ("--") and we call it a String.	

Let us print a string on the console. *Teacher writes code to print "**Hello**" on the console log.*	*The student observes and learns.*

```
  invisibleGround = createSprite(200,190,400,10);
  invisibleGround.visible = false;

  console.log("Hello");

}

function draw() {
  background(180);

  if(keyDown("space") && trex.y>=100) {
    trex.velocityY = -10;
  }

  trex.velocityY = trex.velocityY + 0.8

  if (ground.x < 0){
    ground.x = ground.width/2;
  }

  trex.collide(invisibleGround);
```

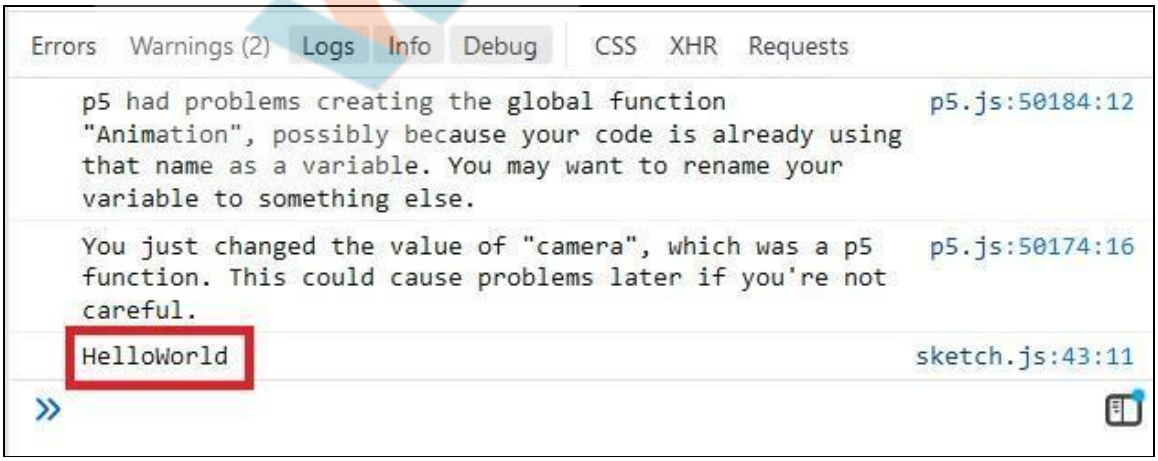
OUTPUT:



Two strings can be joined together using the **'+'** sign, like this: "Hello" + "world" *Teacher writes code to join the strings "**Hello**" and "**World**".* We call it string **concatenation**. Concatenation means joining the two strings together.	*Student observes the output and learns.*

```
  invisibleGround = createSprite(200,190,400,10);
  invisibleGround.visible = false;

  console.log("Hello"+"World");
}

function draw() {
  background(180);

  if(keyDown("space") && trex.y>=100) {
    trex.velocityY = -10;
  }

  trex.velocityY = trex.velocityY + 0.8

  if (ground.x < 0){
    ground.x = ground.width/2;
  }

  trex.collide(invisibleGround);
```

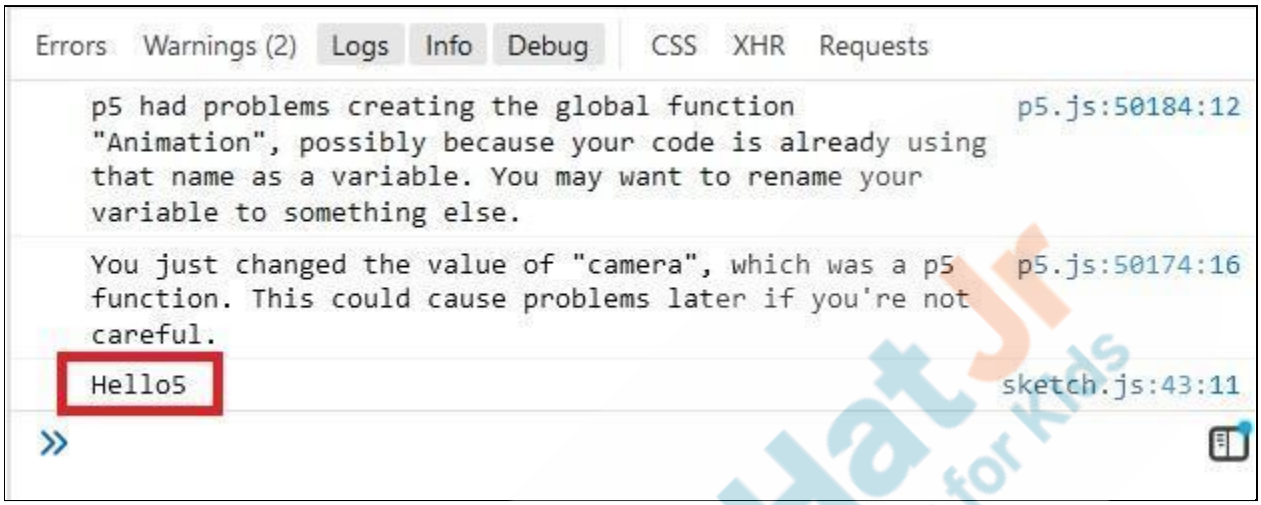
OUTPUT:



A string and a number can also be joined/concatenated together. *The teacher writes code to join the string "**Hello**" and number **5** and display it on the console window.*	*Student observes the output and learns.*

```
  invisibleGround = createSprite(200,190,400,10);
  invisibleGround.visible = false;

  console.log("Hello"+5);

}

function draw() {
  background(180);

  if(keyDown("space") && trex.y>=100) {
    trex.velocityY = -10;
  }

  trex.velocityY = trex.velocityY + 0.8

  if (ground.x < 0){
    ground.x = ground.width/2;
  }

  trex.collide(invisibleGround);
```

OUTPUT:



Teacher starts slideshow **from slide 20- 23**

String concatenation is a very powerful concept and you will be using it to build a scoring system in the game.

Before that, we will learn how to use the '**switch**' statement to spawn different obstacles in the game.

Time to share your screen and start working on this challenge.

Teacher Stops Screen Share

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.**
- **The teacher gets into Fullscreen.**

ACTIVITY

- **Assign lifetime to the spawned objects in the game.**
- **Spawn the obstacles in the game. Use switch statements and randomization together to spawn a variety of obstacles.**
- **Use World.frameCount and Math.round functions to build a simple scoring system.**

Teacher starts slideshow  **from slides 18 to 20**
Refer to speaker notes and follow the instructions on each slide.

Step 3: **Student-Led Activity** Let us create an empty function called spawnObstacles and use it inside the draw function. Right now our empty function tells the computer to do nothing.	*Downloads the Student Activity 1 code and runs in the VS Code editor* *Student writes the code under the teacher's guidance.*

```
  trex.collide(invisibleGround);

  //spawn the clouds
  spawnClouds();

  //spawn obstacles on the ground
  spawnObstacles();

  drawSprites();
}

function spawnObstacles(){

}
```

Look at the animations tab. There are 6 different types of obstacles as obstacle1, obstacle2, obstacle3, obstacle4, obstacle5 and obstacle6.

You need to randomly spawn any of these obstacles on the ground.
Obstacles should appear after a time interval. But creating such a function will be complex. Instead, we can use the frame rate of the game.

We want to display an obstacle after 1 second. Our game has a frame rate of 60. So if we show an obstacle after 60 frames it will be after 1 second only.

Let's create an obstacle sprite every 60 frames or so.

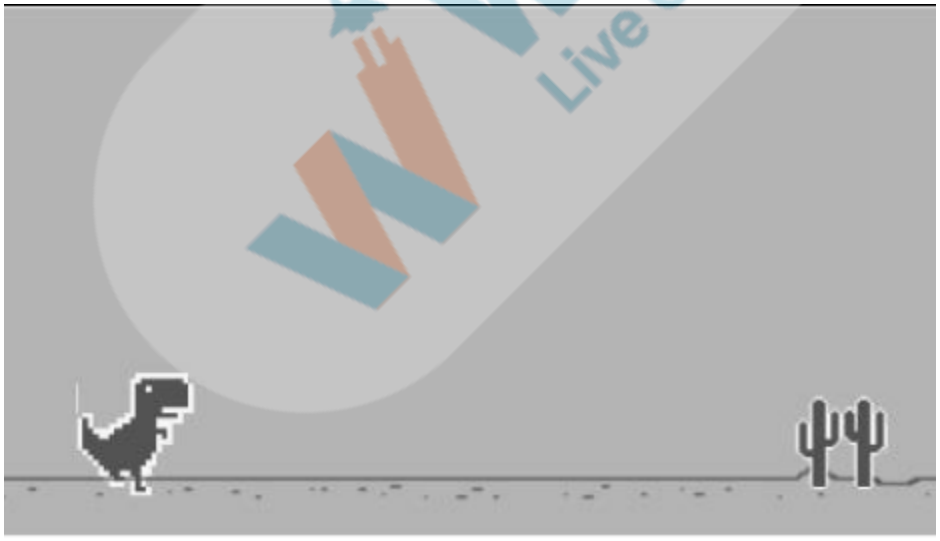
Give the obstacle the same velocity as the ground.
The obstacles need to move with the ground.

Student writes the code under the teacher's guidance.

Student runs the code and observes the output.

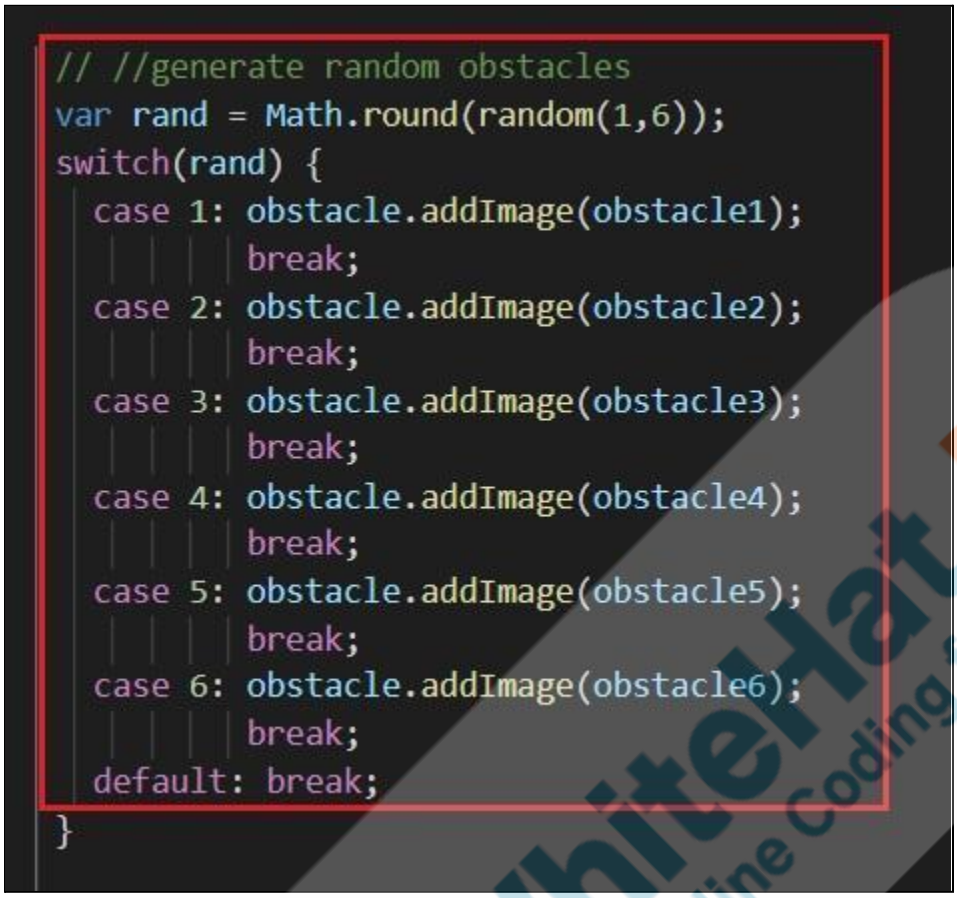
```
  trex.collide(invisibleGround);

  //spawn the clouds
  spawnClouds();

  //spawn obstacles on the ground
  spawnObstacles();

  drawSprites();
}

function spawnObstacles(){
  if (frameCount % 60 === 0){
    var obstacle = createSprite(600,165,10,40);
    obstacle.velocityX = -6;
  }
}
```

Output:



Let us generate and store a random number between 1 to 6. We will use switch case statement to randomly assign different obstacle animations for the obstacle sprites. Since we have 6 obstacles images, every time the draw function runs, a random number will be chosen from 1 to 6, then based on the random number switch case will assign that particular animation to the obstacle sprite. For example if the random number is 1 then, the case 1 will be executed which will assign the obstacle 1 animation.	*Student writes the code under the teacher's guidance.*

```
function preload(){
  trex_running = loadAnimation("trex1.png","trex3.png","trex4.png");
  trex_collided = loadAnimation("trex_collided.png");

  groundImage = loadImage("ground2.png");

  cloudImage = loadImage("cloud.png");

  obstacle1 = loadImage("obstacle1.png");
  obstacle2 = loadImage("obstacle2.png");
  obstacle3 = loadImage("obstacle3.png");
  obstacle4 = loadImage("obstacle4.png");
  obstacle5 = loadImage("obstacle5.png");
  obstacle6 = loadImage("obstacle6.png");

}
```



```
// //generate random obstacles
var rand = Math.round(random(1,6));
switch(rand) {
  case 1: obstacle.addImage(obstacle1);
          break;
  case 2: obstacle.addImage(obstacle2);
          break;
  case 3: obstacle.addImage(obstacle3);
          break;
  case 4: obstacle.addImage(obstacle4);
          break;
  case 5: obstacle.addImage(obstacle5);
          break;
  case 6: obstacle.addImage(obstacle6);
          break;
  default: break;
}
```

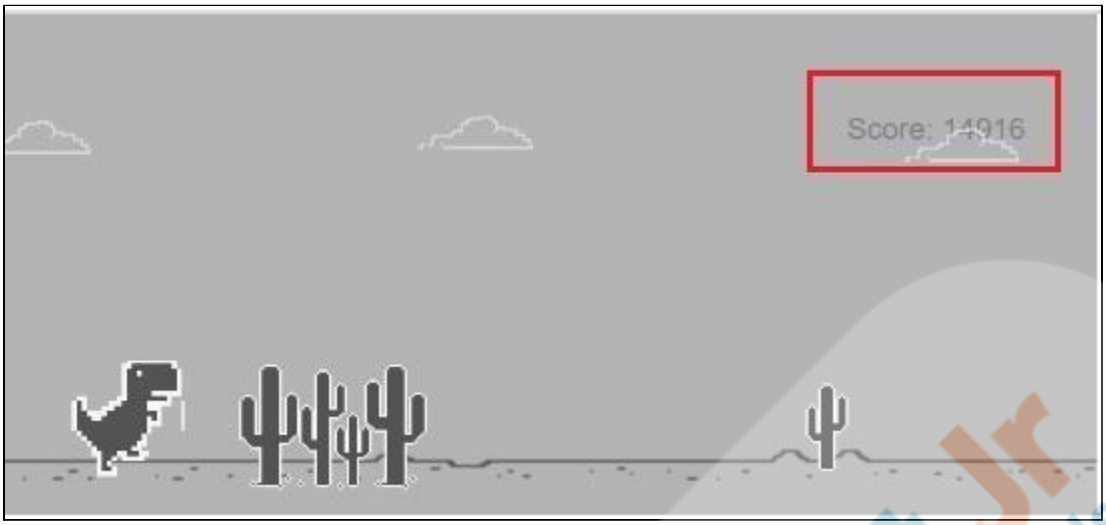
Awesome! Let us scale the obstacles by half and give them a lifetime. What do you think will happen when the code is run? We need to set the scale and lifetime of the obstacles so that we can reduce the lifetime of the obstacle once it moves past the Trex. When lifetime goes to **0**, obstacles will be removed from the memory.	*The student writes code to scale the obstacles and give them a lifetime.* **ESR:** Varied.

```
function spawnObstacles(){
  if (frameCount % 60 === 0){
    var obstacle = createSprite(600,165,10,40);
    obstacle.velocityX = -6;

   // //generate random obstacles
   var rand = Math.round(random(1,6));
   switch(rand) {
     case 1: obstacle.addImage(obstacle1);
             break;
     case 2: obstacle.addImage(obstacle2);
             break;
     case 3: obstacle.addImage(obstacle3);
             break;
     case 4: obstacle.addImage(obstacle4);
             break;
     case 5: obstacle.addImage(obstacle5);
             break;
     case 6: obstacle.addImage(obstacle6);
             break;
     default: break;
   }

   //assign scale and lifetime to the obstacle
   obstacle.scale = 0.5;
   obstacle.lifetime = 300;
  }
}
```

Quickly, let's build a simple scoring system. As the Trex travels more distance in the game, which are the game variables that change? We can use either to keep the score. Let's use **frameCount** as the score. We want to increment the score after 1 second, and we know that our frame is 60 (60 frames per second), so if we divide the frame rate by 60 we will get 1. We will add this to our score, by this way the score will be updated after every 1 second. *Guide the student to display **frameCount** as the score.* ***Note:** Check image below as guidance.*	**ESR:** 1. Time elapsed. 2. Number of frames in the game. *Student writes the code under the teacher's guidance.*

```
  console.log("Hello" + 5);

  score = 0;
}

function draw() {
  background(180);
  text("Score: "+ score, 500,50);
  score = score + Math.round(frameCount/60);
```

Output:

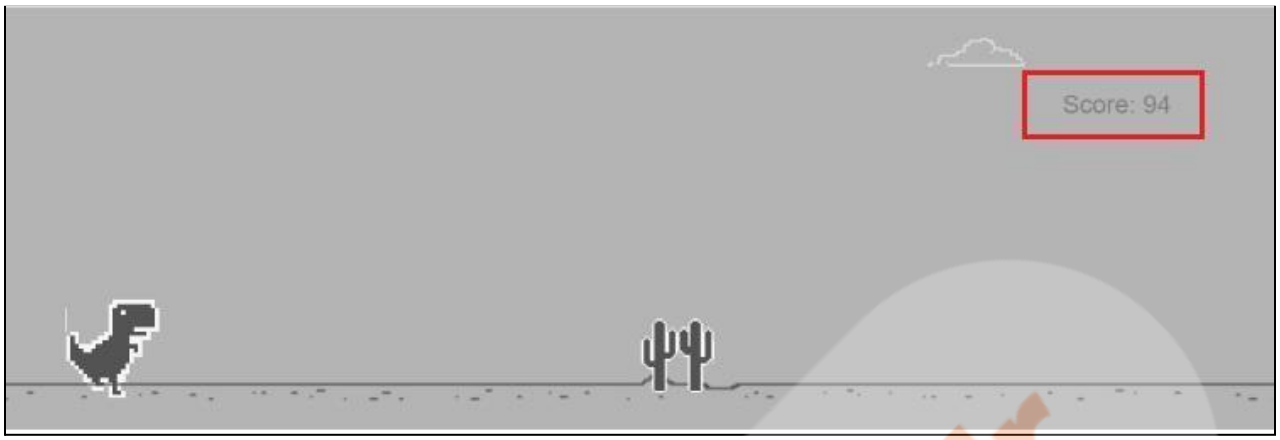


Student writes the code.

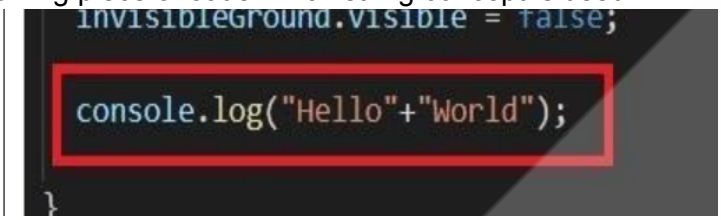
Since we do not want any decimal, we can round off the values using **Math.round()** function. This function rounds the value to the closest integer, for example if we have the values as 9.7, **Math.round()** will make it as 10.

```
  console.log("Hello" + 5);

  score = 0;
}

function draw() {
  background(180);
  text("Score: "+ score, 500,50);
  score = score + Math.round(frameCount/60);

  if(keyDown("space")&& trex.y >= 362) {
    trex.velocityY = -13;
  }
```

Output:



Teacher Guides Student to Stop Screen Share	
WRAP UP SESSION - 5 Mins	
Teacher starts slideshow  **from slide 21 to slide 32**	
Activity details	**Solution/Guidelines**
Run the presentation from slide 21 to slide 32. **Following are the warm up session deliverables:** • **Explain the facts and trivias** • **Next class challenge** • **Project for the day** • **Additional Activity**	Guide the student to develop the project and share with us.
Quiz time - Click on in-class quiz	
Question	**Answer**
Which of the following statements is used to switch the response based on the value? A. If condition B. Switch case statement C. For loop D. Nested if	**B**

Which of the following commands is used to solve memory leak issues ? A. Sprite.lifetime B. Sprite.depth C. Sprite.destroy() D. A and C both	**D**
In the following piece of code which string concept is used?  A. Concatenation B. Displaying text C. Converting string to integer D. None of the above	**A**
End the quiz panel	
FEEDBACK • **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.**	
You get Hats Off for your excellent work! We have most of the things ready for our game. In the next class we will write code to build collisions with the obstacles and use game states which we learned in the last class.	*Make sure you have given at least 2 Hats Off during the class for:* 

You might want to try it on your own too. Trying to change the logic in the code is a very good way to enhance understanding. If you try to change the parameters you will be able to build a very strong understanding of them. How are you feeling about the class?	**ESR:** Varied.
Great! You can also spend some time adjusting the mechanics of the game like speed, gravity etc. **Note**: Help the student change gravity, speed and other variables in the game to see how it affects the game. We will see each other again in the next class. The trex will be waiting for you.	
*** This Project will take only 30 mins to complete. Motivate students to try and finish it immediately after the class.** **Project Overview** **BALLOON BUSTER - 1** **Goal of the Project:** Today, you have learned how to correct memory leak problems in code. You have used string concatenation to display a scoring system. In this project, you will have to practice and apply what you have learned in the class and add lifetime to all balloons according to their velocity. Also, display the scoring system for the game using string concatenation	**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.** Students engage with the teacher over the project.

** This is a continuation of Project 11, so make sure to complete the C12 Project before doing this project. ** **Story**: So far you have helped Meera a great deal to design a game of bursting balloons with an arrow. Now you are just a few steps away to complete the game. You need to add a scoring system today using string concatenation. I am very excited to see your project solution and I know you will do really well. Bye Bye!	
Teacher ends slideshow 	
Teacher Clicks 	
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?	*Student uses the markdown editor to write her/his reflection as a reflection journal.*

Activity	Activity Name	Links
Teacher Activity 1	Template link	https://github.com/pro-whitehatjr/Pro-c13-TA1_template
Teacher Activity 2	Solution link	https://github.com/pro-whitehatjr/c13_TA1-Solution
Teacher Activity 3	Trex stage 3	https://github.com/pro-whitehatjr/C13_trex3
Teacher Activity 4	Reference	https://github.com/pro-whitehatjr/C13_full_game
Student Activity 1	Trex stage 3.5	https://github.com/pro-whitehatjr/C13_trex3.5
Teacher Reference visual aid link	Visual aid link	https://curriculum.whitehatjr.com/Visual+Project+Asset/PRO_VD/BJFC-PRO-V3-C13-With+Cues.html
Teacher Reference In-class quiz	In-class quiz	https://s3-whjr-curriculum-uploads.whjr.online/4a3ed0d9-3679-4906-a567-3ac1fd0ba55c.pdf
Project Solution	BALLOON BUSTER - 1	https://github.com/pro-whitehatjr/Project_C13_Balloon_Buster_1