Topic	CODE DEBUGGING AND CODE INDENTATION	
Class Description	**Students learn to indent code to make it more readable. Students use the method of logging on the console to debug the program.**	
Class	**PRO-C10**	
Class time	**45 mins**	
Goal	• Indent the code correctly to make it more readable. • Identify an additional condition needed in the program to stop the trex from jumping again while it is in the air. • Create an invisible ground sprite to make the trex run below the ground.	

Resources Required	• Teacher Resources
	  ○ P5 editor login
	  ○ Laptop with internet connectivity
	  ○ Earphones with mic
	  ○ Notebook and pen
	• Student Resources
	  ○ P5 editor login
	  ○ Laptop with internet connectivity
	  ○ Earphones with mic
	  ○ Notebook and pen	

Class structure	**Warm Up - Slide show option** **Teacher-led Activity** **Student-led Activity** **Wrap Up - Slide show option**	**5 mins** **10 mins** **20 mins** **5 mins**
WARM UP SESSION - 15mins		

Teacher starts slideshow  **from slides 1 to 31** Refer to speaker notes and follow the instructions on each slide.	
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 6.** **Following are the warm up session deliverables:** ● Connecting students to the previous class.	**ESR**: Hi, thanks, yes I am excited about it! Click on the slide show tab and present the slides.
QnA Session	
Question	**Answer**
Which of the following options can be used to reset the ground position? A. Reset the ground back to the center if the **y-position** of the ground becomes less than 0. We can bring it back to its original position. This way, the ground will always be there. B. Reset the ground back to the center if the **x-position** of the ground becomes less than 0. We can bring it back to its original position. This way, the ground will always be there. C. Reset the ground back to the center if the **y-position** of the ground becomes greater than 0. We can bring it back to its original position. This way, the ground will always be there. D. Reset the ground back to the center if the **x-position** of the ground becomes greater than 0. We can bring it back to its original position. This way, the ground will always be there.	**B**
	B

We can scale the Trex by half by? A. Trex.scale=0.05 B. Trex.scale=0.5 C. Trex.scale=50 D. Trex.scale=0.1	
Continue the warm up session	
Activity details	**Solution/Guidelines**
Run the presentation from slide 7 to slide 31 to set the problem statement. ● Explaining the debugging process through real life connections. ● Bugs encountered in the Trex game. ● Introduce the importance of indentation and spacing through an example. ● Steps to write and run the code.	Narrate the story by using hand gestures and voice modulation methods to bring in more interest in students.
Teacher ends slideshow 	
TEACHER-LED ACTIVITY - 8mins	
Teacher Initiates Screen Share	

Activity details		**Solution/Guidelines**
Step 2: Teacher-led Activity (10 mins)	*Teacher opens **Teacher Activity 1**.* "This is the code from the last class. Do you see any problems with this code?"	**ESR:** There is no spacing between the lines of the code. They are all together and difficult to read.
	"Yes, computers don't mind or read spaces. But it is important to give spaces in your code to make them	*The student listens and learns.*

	easily readable. Remember, you want other programmers to easily read your code. Giving proper spaces in your code makes your code easily readable. Let's try to add some space in your code."	
	"We try to leave a line after a meaningful block of code." Can you tell which code lines together make a meaningful block of code? Teacher leaves lines after each meaningful block of instruction.	*The student observes, comments and learns.*

```
function setup() {
createCanvas(600, 200);

//create a trex sprite
trex = createSprite(50,160,20,50);
trex.addAnimation("running", trex_running);
trex.scale = 0.5;

//create a ground sprite
ground = createSprite(200,180,400,20);
ground.addImage("ground",groundImage);
ground.x = ground.width /2;
ground.velocityX = -4;

}

function draw() {
background(220);

//jump when the space button is pressed
if (keyDown("space")) {
  trex.velocityY = -10;
}

trex.velocityY = trex.velocityY + 0.8

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

trex.collide(ground);
```

	There is no fixed rule to leaving these spaces. It is just like leaving spaces between paragraphs when you write a story. But do you see how easy it is to read and understand the code now?	**ESR:** Yes
	We also add some code indentation to lines to show that they are contained inside another block of code.	*The student observes and learns.*

	Let me show you how. For example: lines 30 to 47 are contained inside the function draw(). We show it by indenting these lines by adding some space in front of these lines. This spacing should be consistent. *Teacher selects lines 30 to 47 and presses TAB.*	

	This is called code indentation. Similarly, the lines of code inside **if** blocks or **for** blocks need to be indented as well.	The student observes and learns.

	Code indentation helps us understand the program structure easily. It also makes us less likely to make mistakes while typing out text - like missing out on closing curly brackets.	*The student listens and learns.*

	Ok now it's time to work on solving the two bugs we had in our program. Bugs are parts of the program that do not work as we want. We have two such bugs that we will solve for today. - the dinosaur jumping in mid air when the space key is pressed. What do we want instead?	**ESR:** We want the dinosaur to be able to jump only when it is touching the ground. It should be able to jump again only when it falls back on the ground.
	Great! Our second bug is that our dinosaur is running a little over the ground. What do we want instead?	**ESR:** We want it to run a little below the ground.
	Ok let's fix these.	
Teacher Stops Screen Share		
STUDENT-LED ACTIVITY - 10 mins		
Teacher starts slideshow  **Slide 32 (Only one slide.)**		
Activity details		**Solution/Guidelines**
Run the presentation slide 32 only. ● Identify and add an additional condition so that the Trex jumps only when in contact with the ground. ● Create an invisible ground sprite which is below the ground and on which the Trex dinosaur is supported.		Guide the student to develop the project and share with us.

Teacher ends slideshow 		
• **Ask Student to press ESC key to come back to panel** • **Guide Student to start Screen Share** • **Teacher gets into Fullscreen**		
ACTIVITY • **Identify and add an additional condition so that the Trex jumps only when in contact with the ground.** • **Create an invisible ground sprite which is below the ground and on which the Trex dinosaur is supported.**		
Step 3: Student-Led Activity (10 mins)	Ok. Quickly fire up your activity and indent your code. *Teacher helps the student to properly indent their code.*	*Student opens Student Activity 1 and presses File> Duplicate.*
	Let us first fix the second bug. The Trex right now is supported by the ground sprite. Collision with the ground sprite is not letting the Trex fall off the ground. As our Trex looks like it's running a little above the ground to fix this we'll create an invisible ground just below the original ground and make our trex run on the invisible ground so that it looks like the Trex is running on the ground. Let us create an invisible ground sprite just below this ground. We want to do this so that rather than being supported by the ground and being above the ground- Trex gets	*Student creates an invisible ground Sprite.*

	supported by an invisible ground just below the actual ground. Can you create another ground sprite just below the first ground and make it cover the entire width of the screen. For fun's sake, let us call it invisibleGround. *Guide the student to create an invisible ground Sprite.*	

```
  //create a trex sprite
  trex = createSprite(50,160,20,50);
  trex.addAnimation("running", trex_running);
  trex.scale = 0.5;

  //create a ground sprite
  ground = createSprite(200,180,400,20);
  ground.addImage("ground",groundImage);
  ground.x = ground.width /2;
  ground.velocityX = -4;

  //creating invisible ground
  invisibleGround = createSprite(200,190,400,10);
  invisibleGround.visible = false;

}

function draw() {
  //set background color
  background(220);

```

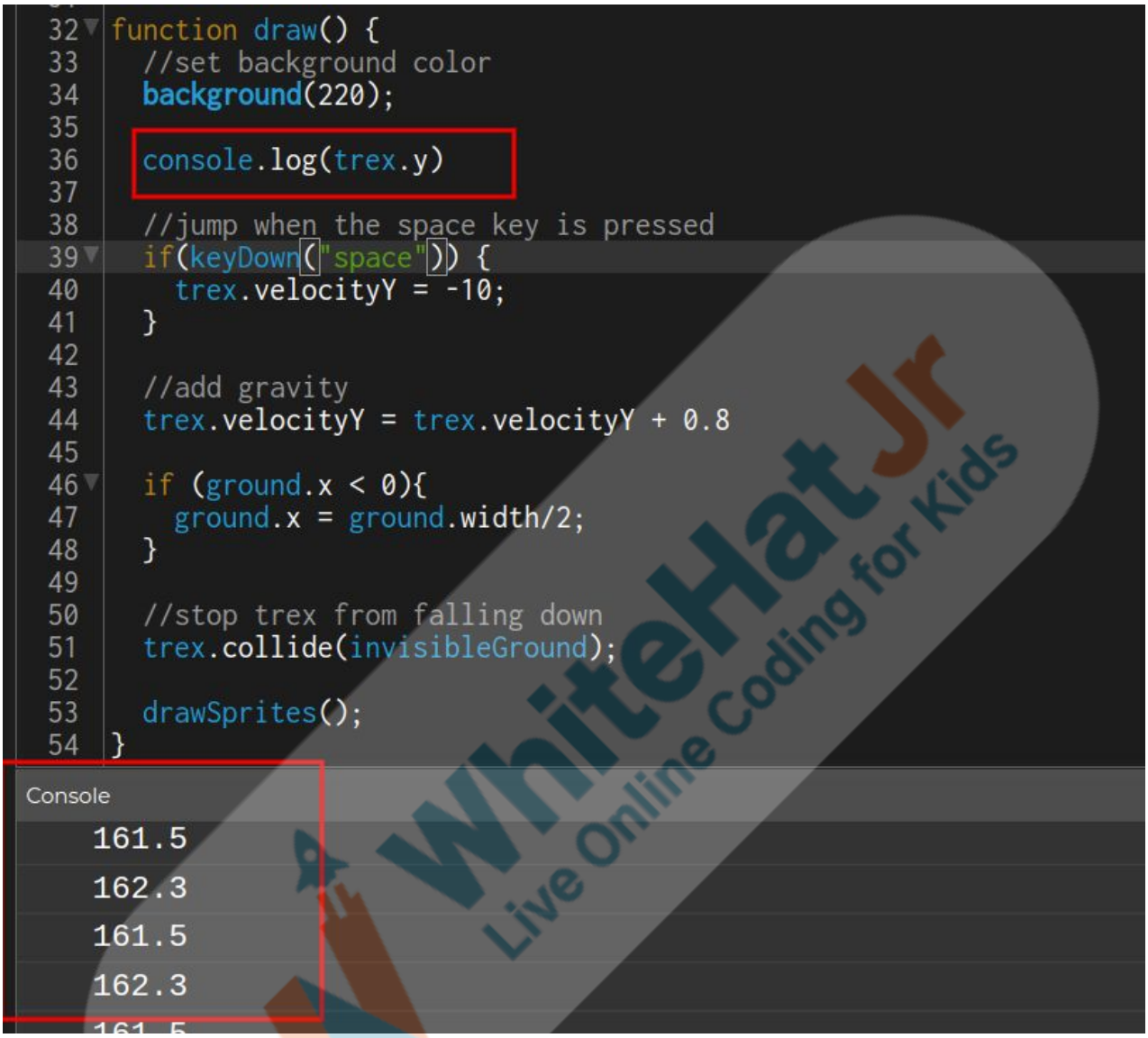
	Instead of supporting the trex on the ground, let us collide it with the invisible ground.	*Student modifies trex.collide(ground) to trex.collide(invisibleGround)*

```
31
32  function draw() {
33    //set background color
34    background(220);
35
36    console.log(trex.y)
37
38    //jump when the space key is pressed
39    if(keyDown("space")) {
40      trex.velocityY = -10;
41    }
42
43    //add gravity
44    trex.velocityY = trex.velocityY + 0.8
45
46    if (ground.x < 0){
47      ground.x = ground.width/2;
48    }
49
50    //stop trex from falling down
51    trex.collide(invisibleGround);
52
53    drawSprites();
54  }
```

	Now for the magic! Let us make this ground sprite invisible. There is an instruction in the toolbox sprite.visible. You need to make it false to make the ground invisible. By saying 'sprite.visible = false' , we are asking the computer to NOT make this sprite visible.	*Student writes **invisibleGround.visible = false** to make the ground invisible.* *Student runs the code to see the output.*

Add the following line of code anywhere outside the function draw() and after creating the invisibleGround Sprite: **invisibleGround.visible = false;** **Note- Click on the canvas where you see the output to use the keys to make the trex jump**		
	We have the trex running on the ground now!!	*Student amazed :)*
		
	Let us try to fix the other bug which is …….	**ESR:** Make the dinosaur jump only when it is on the ground.
	When does the dinosaur jump now?	**ESR:** When the space key is pressed.

	We want to make it jump when the space key is pressed and when it is on the ground. Right?	**ESR:** Yes
	What do we need to do ?	**ESR:** Add an additional if condition?
	Right. Let us try to log the current y position of the Trex when it is running on the ground. *Guide the student to log trex.y on the console window.*	*Student writes code to **log** trex.y on the console window.*



```
function draw() {
  //set background color
  background(220);

  console.log(trex.y)

  //jump when the space key is pressed
  if(keyDown("space")) {
    trex.velocityY = -10;
  }

  //add gravity
  trex.velocityY = trex.velocityY + 0.8

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

  //stop trex from falling down
  trex.collide(invisibleGround);

  drawSprites();
}
```

Console

```
161.5
162.3
161.5
162.3
```

	What do you see?	**ESR:** The y position of the trex changes between 362 and 362.8

	Right. and when it jumps what will happen to trex.y?	*Student runs the code and makes the trex jump to see the change in the trex.y in the console*. **ESR:** trex.y reduces when the trex jumps.
	So we want the trex to jump only when it is on the ground , that is, only when trex.y >= 100. How can we do that?	**ESR:** By adding an additional condition inside the if block where we make the trex jump.
	Let's do that. *Teacher guides the student to write the additional condition inside 'If block'*	*Student writes the code and runs the program.* *The student can press space repeatedly to see if the program works as expected.*

```
function draw() {
  //set background color
  background(220);

  console.log(trex.y)

  //jump when the space key is pressed
  if(keyDown("space") && trex.y >= 100) {
    trex.velocityY = -10;
  }

  //add gravity
  trex.velocityY = trex.velocityY + 0.8

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

  //stop trex from falling down
  trex.collide(invisibleGround);

  drawSprites();
}
```

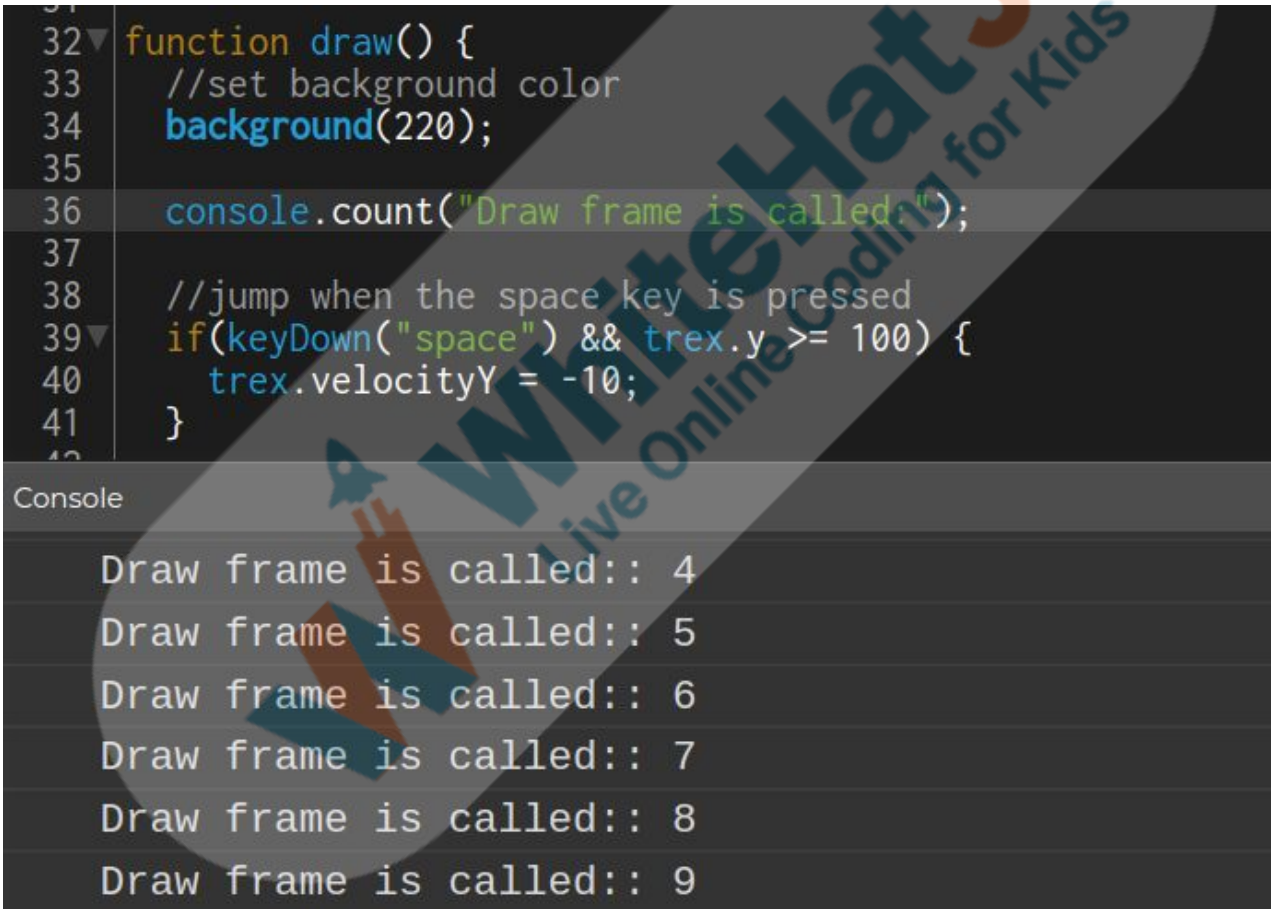
Teacher Guides Student to Stop Screen Share

Quiz time - Click on in-class quiz

Question	Answer
Debugging is _____ A. finding and fixing the bugs. B. introducing bugs in code. C. running the code. D. writing the code.	**A**
Code indentation is done A. so that the program runs without error.	D

B. to resolve the errors. C. to find the errors. D. to make the code easily readable.	
Ground sprite can be made invisible by setting sprite A. sprite.visible=true; B. sprite.visible=false; C. sprite.visible=1; D. sprite.visible=0;	B
End the quiz panel	
WRAP UP SESSION - 5 Mins	
Teacher starts slideshow  **Slide 33-39**	
Activity details	**Solution/Guidelines**
Run the presentation from slide 33 to slide 39. **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.
Teacher ends slideshow 	
Next class we will look at how to create floating clouds at different heights. Thank you <Friend name> for joining us in this class. I hope to see you again sometime. Looking forward to our next class.	-

Project Overview	**Note:** This is a tiered project with multiple tasks. All students must do the main task. The main task is very similar to the projects that are already live. Each tiered project has two or more additional tasks which are optional. **RUN JOHN RUN** **Goal of the Project:** In this class we have learned how to indent code and use the console display live position of an object. Using visible properties of ground we made ground invisible. In this project, you have to create a vertically moving background and an animated boy sprite. **Story:** John visited his best friend's home. He loved the Running Surfers game which his friend was playing on his mobile. When he came back home, he tried to play that game on his mobile but his mobile was not supporting that Running Surfers game. So he decided to build a computer game similar to the actual subway surfers game. Can you help Lucifer design the game I am very excited to see your project solution and I know you will do really well.	*Students engage with the teacher over the project.*

	Bye Bye!	
Teacher Clicks 		
ADDITIONAL ACTIVITIES		
• **Ask Student to press ESC key to come back to the panel** • **Guide Student to start Screen Share** • **Teacher gets into Fullscreen**		
		
	You can also use the console to find out how much time it takes your program to run. We use console.time() to start keeping log of the time and	*The student learns how to use console.time and console.timeEnd to log the time taken by the program to run.*

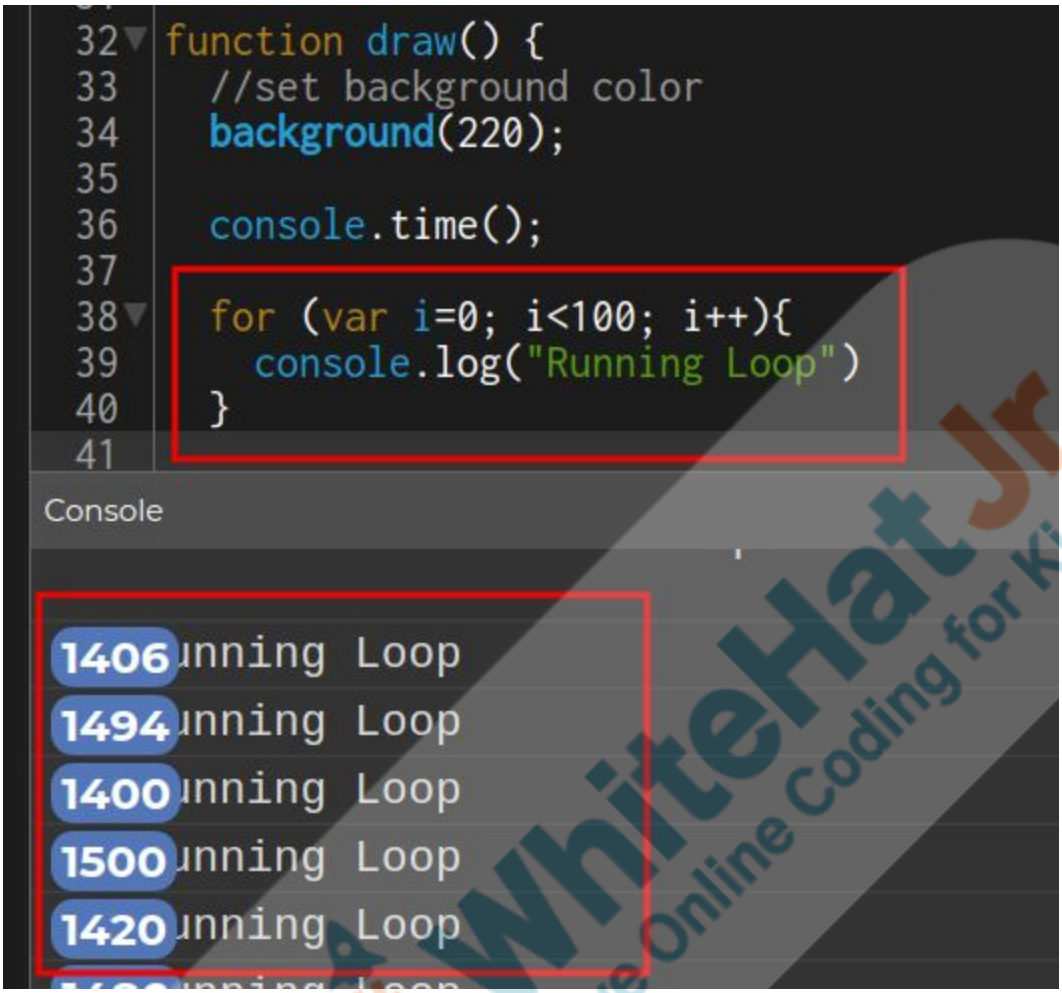
	console.timeEnd() to stop and print the time on the console. This is used by programmers when they want to optimize and reduce the time taken by their program to run. *Teacher shows how to use console.time() and console.timeEnd() to measure the time taken by the draw function to run.*	

console.time() when the draw function starts

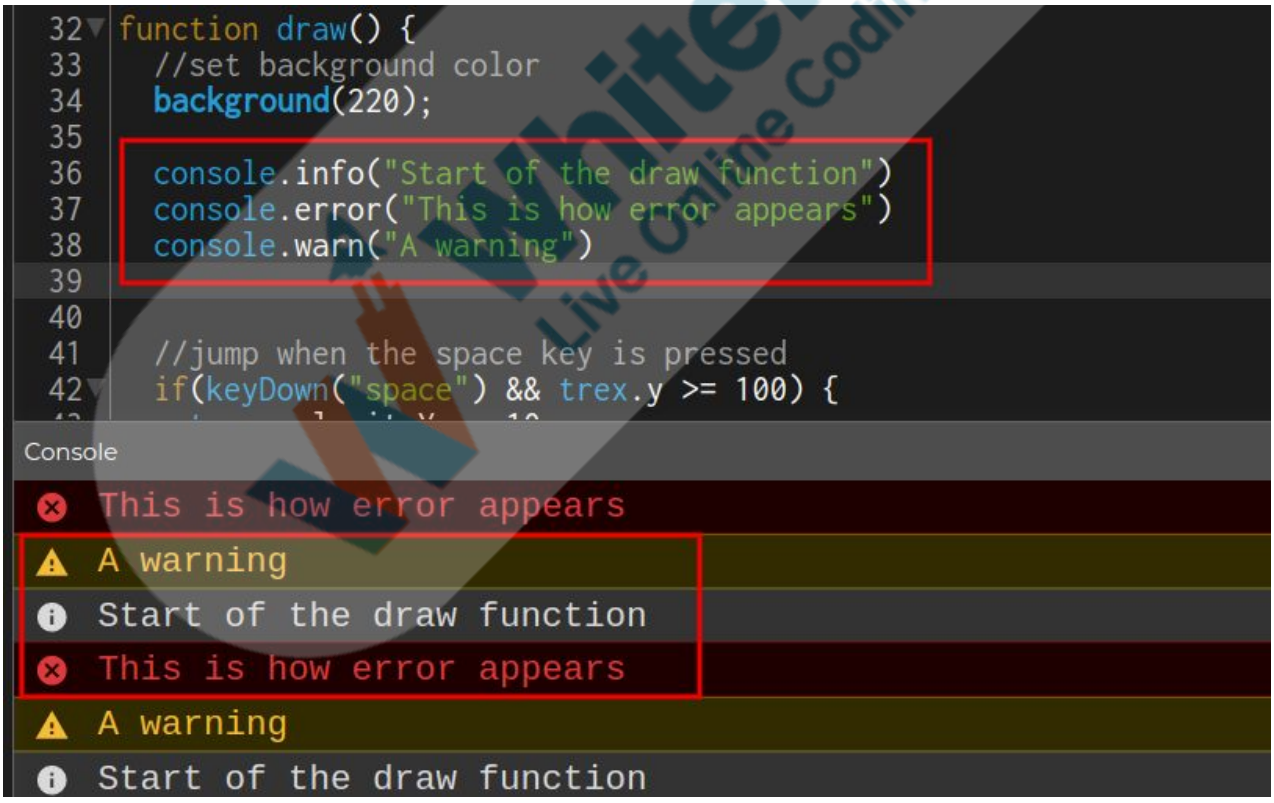
console.timeEnd() when the draw function ends

	Similarly, you can also find out how long it takes for function setup or function preload to run before your game can start. **Note:** Observe how it takes different time each time the draw function runs. The variation is because your computer's processing speed depends on a lot of factors like - how heated your computer chips are, what are the other things your computer is	*Student writes a simple for-loop inside the function draw() and observes the change in execution time of the draw function.* *The student observes the lag in the game - where every character slows down and gives an impression of the game hanging.*

	doing - for example what else is happening on your browser etc. Let us write a simple for loop inside function draw. Log anything inside the for loop and check if the execution time of the draw function changes. Also, observe the effect of this on your game. *Ask the student to explain the reason behind the lag in the game.*	**ESR:** Every frame in the game is rendered (drawn) each time the draw function gets called. The lag in the game is because it takes longer for the next frame to render.



	Good! Our goal should always be to write programs which run in the least time possible. There are other ways you can use the console. console.log() is used to print a simple message. You can use console.warn() to print a warning. The warning message is formatted differently.	*Student experiments with different types of console messages.*

	Similarly, you can use console.error() to print an error(). the error message is formatted differently. You can also use console.info() to print any information. *Teacher shows how to print information, errors and warnings on the console.* These are especially helpful when you are working on a big project with several other developers. You want your program to be meaningful for them.	
		
	Encourage the student to write reflection notes in their reflection	*Student uses the markdown editor to write her/his*

	journal using markdown. Use these as guiding questions: • What happened today? • Describe what happened. • 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?	*reflection as a reflection journal.*

Activity	Activity Name	Links
Teacher Activity 1	Unindented code	https://editor.p5js.org/Abhijeet/sketches/zbK8rf0uz
Teacher Activity 2	Finished code for reference	https://editor.p5js.org/Abhijeet/sketches/NNVga4nFL
Student Activity 1	Unindented code	https://editor.p5js.org/Abhijeet/sketches/zbK8rf0uz
Teacher Reference visual aid link	Visual aid link	https://curriculum.whitehatjr.com/Visual+Project+Asset/PRO_VD/PRO-C10-withcues.html
Teacher Reference In-class quiz	In-class quiz	https://curriculum.whitehatjr.com/Visual+Project+Asset/PRO_VD/PRO-C10.docx.pdf