Topic	SPRITE OBJECTS
Class Description	The student is introduced to JavaScript variables and sprite objects. The student learns to experiment with predefined sprite object properties and functions, to animate the sprite as a ball on the canvas.
Class	PRO-C1
Class time	50 mins
Goal	● Create a sprite object. ● Access predefined sprite properties and functions.

Resources Required	● Teacher Resources:
	  ○ Code.org login credentials
	  ○ Laptop with internet connectivity
	  ○ Earphones with mic
	  ○ Notebook and pen
	  ○ Smartphone
	
	● Student Resources:
	  ○ Code.org login
	  ○ Laptop with internet connectivity
	  ○ Earphones with mic
	  ○ Notebook and pen

Class structure	Warm-Up Slides 5 mins Teacher-Led Activity 1 10 mins Student-Led Activity 1 5 mins Game Development Slides 5 mins Teacher-Led Activity 2 10 mins Student-Led Activity 2 10 mins Wrap-Up Slides 5 mins

WARM-UP SESSION - 5 mins

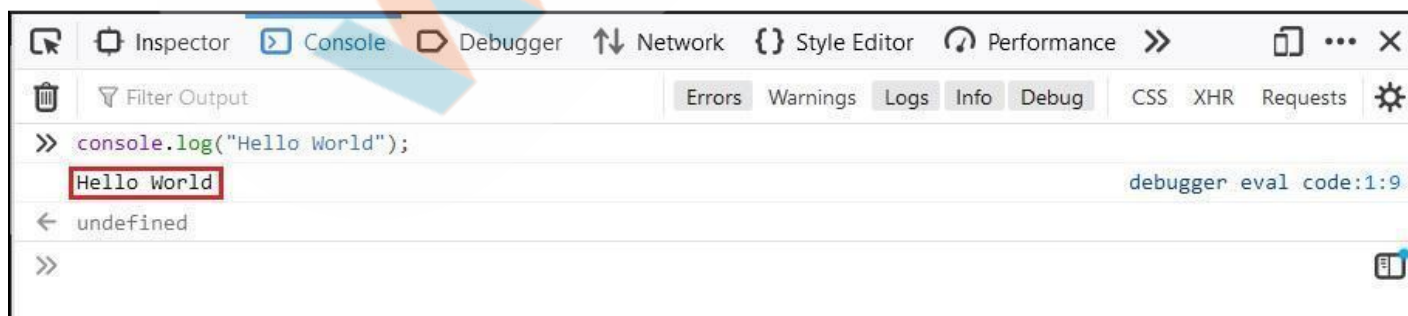|		
Teacher starts slideshow 📊 and presents slides 1 to 10 Refer to the speaker notes and follow the instructions on each slide.		

Teacher Action	Student Action
Hi, it's so good to see you after the trial class! How have you been? So, this is going to be the first class of the first module. Are you excited to learn something new? **The following are the warm-up session deliverables:** ● Connect the student to the Trial Class. ● Variables and Objects ● Warm-Up Quiz Session	**ESR**: Varied Response. Click on the slide show tab and present the slides.

Q & A Session - Click on In-Class Quiz	
Question	**Answer**
| Consider the given code block of the game, Pong. Which of the following instructions are used for drawing the computer paddle?<br><br>```
1  function setup() {
2    createCanvas(400, 400);
3  }
4
5  function draw() {
6    background("white");
7    rect(390, mouseY, 10, 70);
8    rect(0,160,10,70);
9    rect(185,185,10,10);
10
11 }
```<br><br>A. rect (390, mouseY, 10, 70);<br>B. rect (185, 185, 10, 10);<br>**C. rect (0, 160, 10, 70);** | **C** |

D. createCanvas (400, 400);	
| What does the number 390 in line 7 represent in the given code block? <br><br> ```
1 function setup() {
2   createCanvas(400, 400);
3 }
4
5 function draw() {
6   background("white");
7   rect(390, mouseY, 10, 70);
8   rect(0,160,10,70);
9   rect(185,185,10,10);
10
11 }
``` <br><br> A. xPosition of the computerPadddle <br> B. xPosition of the playerPaddle <br> C. yPosition of the computerPaddle <br> D. yPosition of the playerPaddle | B |

Continue the WARM-UP SESSION	
Activity Details	**Solution/Guidelines**
Run the presentation from slide 4 to slide 10 to set the problem statement. **The following are the warm-up session deliverables:** ● Introduce students to the coding environment - Workspace, blocks, and output. ● Steps to write and run the code. ● Introduce the concepts of Teacher-Led Activity.	Narrate the slides by using hand gestures and voice modulation methods to bring in more student interest.

Teacher ends slideshow
TEACHER-LED ACTIVITY 1 - 10 mins
Teacher Initiates Screen Share
GOALS & CHALLENGES

- **Introduction to Game Lab & Console.**
- **Introduction to Variables.**
- **Practice Variable Declaration, Initialization and Assignment Operator.**

Teacher-Led Activity (10 mins)

Teacher Action	Student Action
All modern browsers like Chrome, Firefox, etc. have a web console that helps us interact with a web page by writing JavaScript code and running them. For now, let's understand how we can instruct computers to send a message on the console. There is a function called **console.log()**. It is used to write messages to these consoles. Let me show you how to write a message "**Hello world**" on your browser. 1. Open your browser, **right-click** anywhere and click on **Inspect**. (shortcut F12). 2. On the developer tools, click the **Console** tab. Then, write code **console.log("Hello world");** and press **Enter**. *The teacher should write the '**name of the child**' instead of '**world**' to customize for the child. console.log("Hello <<student_name>>");* Did you see the output on the console?	*The student looks at the browser console and verifies the output of the code written by the teacher.*

Though we can write these small instructions for computers on a web console, they are not a good choice if you have to code games which	

require too many lines of code. Every time you have to test your code, you will have to paste your code again.

That's why for writing bigger programs and coding games, we need a platform where we can easily **edit, save and test our code** multiple times.

For this, we will use a platform called "**Game Lab**" in **code.org**. Before we start, let's quickly review the **Game Lab** platform.

The teacher logs into code.org, opens [Teacher Activity 1](#) and presses **Remix***.*
The **left panel** is where the computer will show output for the code that we write.
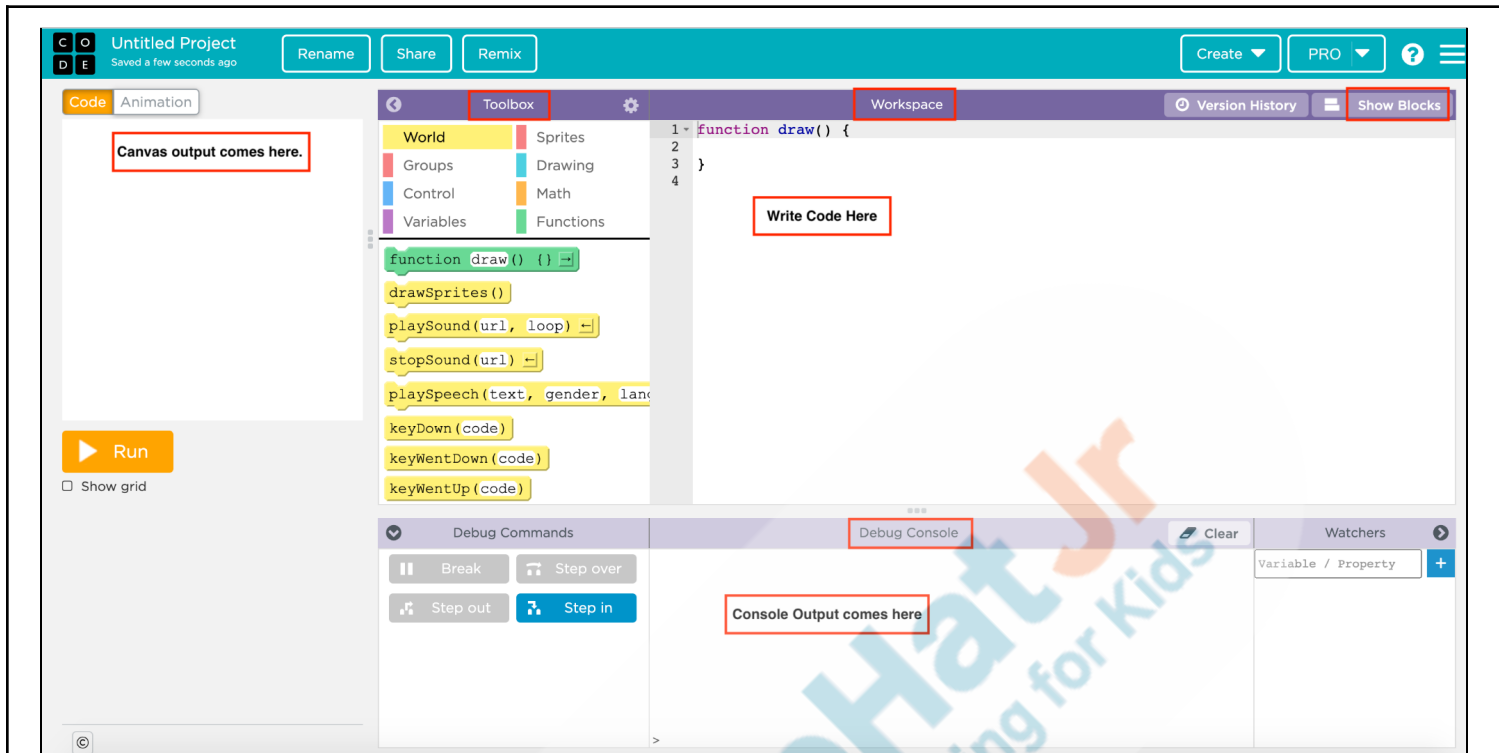
The **Toolbox** contains the list of instructions we can give to the computer. We will be exploring it in more detail later during the class.

The right panel is the **Workspace** where we will be writing our code. You can switch the workspace into "**Block mode**" by clicking "**Show Blocks**" and "**Text mode**" by clicking "**Show Text**". In the **"Block mode"** you can code using Blocks whereas in the **"Text Mode"** you will use JavaScript language to code. You can work in either mode - both result in the same output.

Debug Console is where you will see all the console messages which can be string messages or variable values. **Debug console** is very useful for programmers to find the bugs in the program, but the user of your game will not be able to see these console messages.

Note: *The teacher and student should work in the* **Text mode***. Lesson plans are designed to cater to the text-based programming mode.*
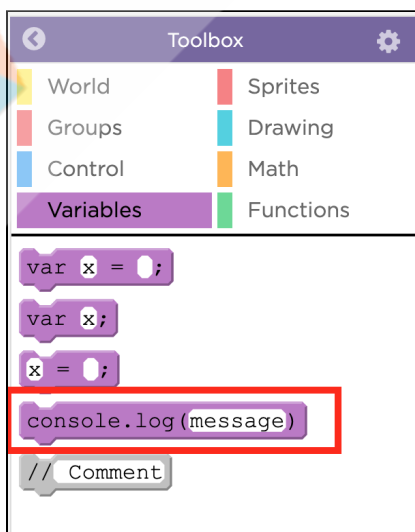
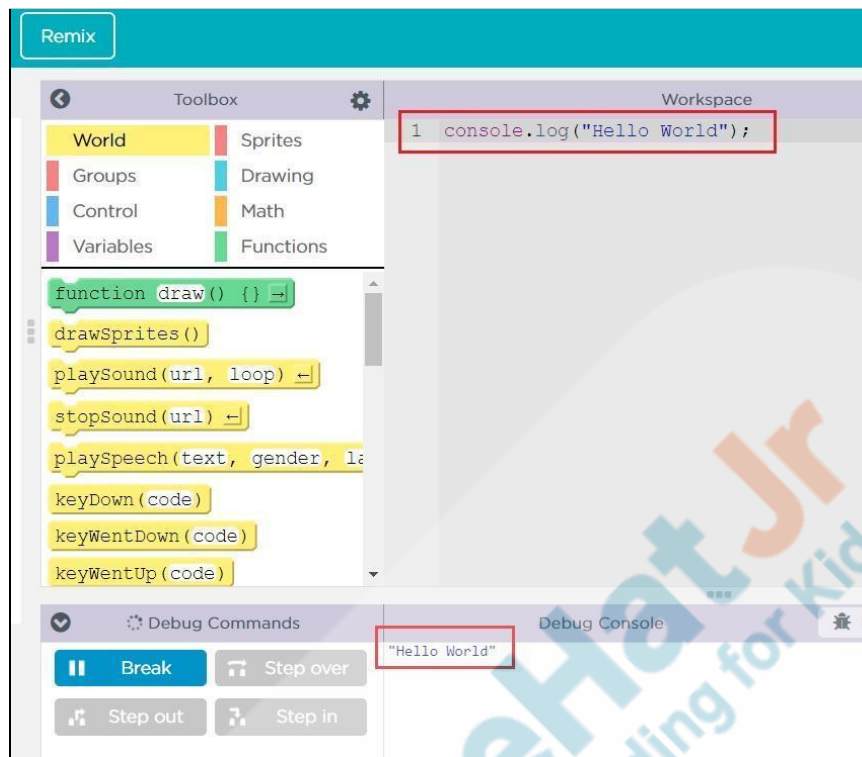The student observes the Game Lab platform.

Let's find out if we have a **console.log()** function in the **Toolbox**.
Let's check inside the **Variable** tab. Can you see it?

The teacher drags it and drops it in the workplace.

Update the message block to "**Hello World**" (Use "**Hello <<student_name>>**" in live class) and click on **Run**.

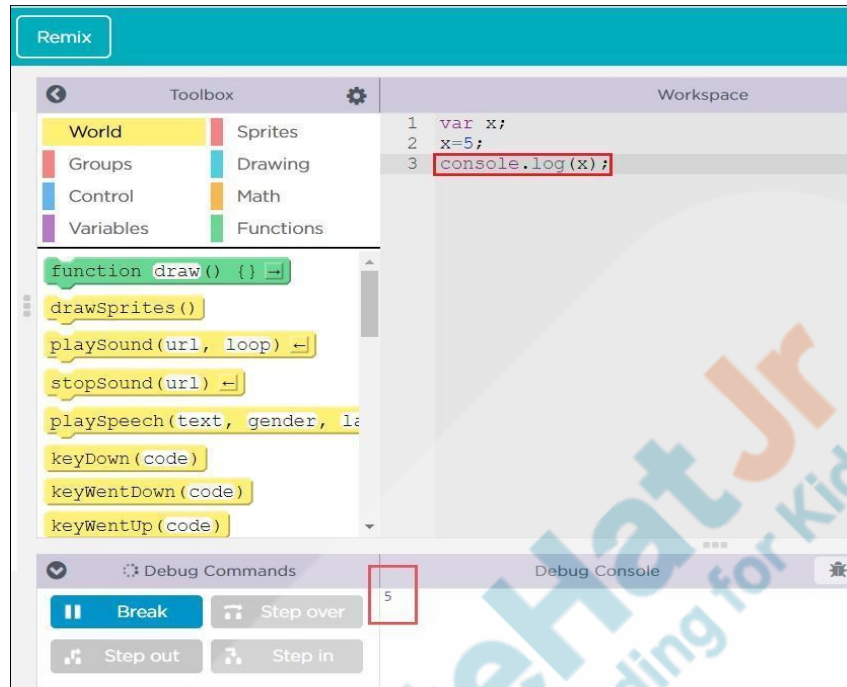|	
Did you see the output in the console?	**ESR**: Yes.
Great! Now that we know the Game Lab interface, let us learn how to create a variable and assign values to it in JavaScript programming.	
Can you look for an instruction to create a variable in the toolbox? *The teacher drags it to the workplace.*	**ESR**: **var x;**
How will you assign value to a variable in Mathematics? Suppose you have to assign value 5 to x. How will you do it?	**ESR:** x=5
Correct. Same works in coding as well. Can you help me find an instruction to give value to the variable x? *The teacher declares a variable and assigns the value '5' to it.*	**ESR**: x= ?
In JavaScript, we use the **var** keyword to declare a variable. As soon as you declare a variable, the computer assigns memory to it. '**x**' is the name of a variable here. We use an assignment operator " **=** " to assign a value to a variable. Let's now print its value on the console.	**ESR**: The student observes the screen.

The teacher prints it on the console and runs the program.

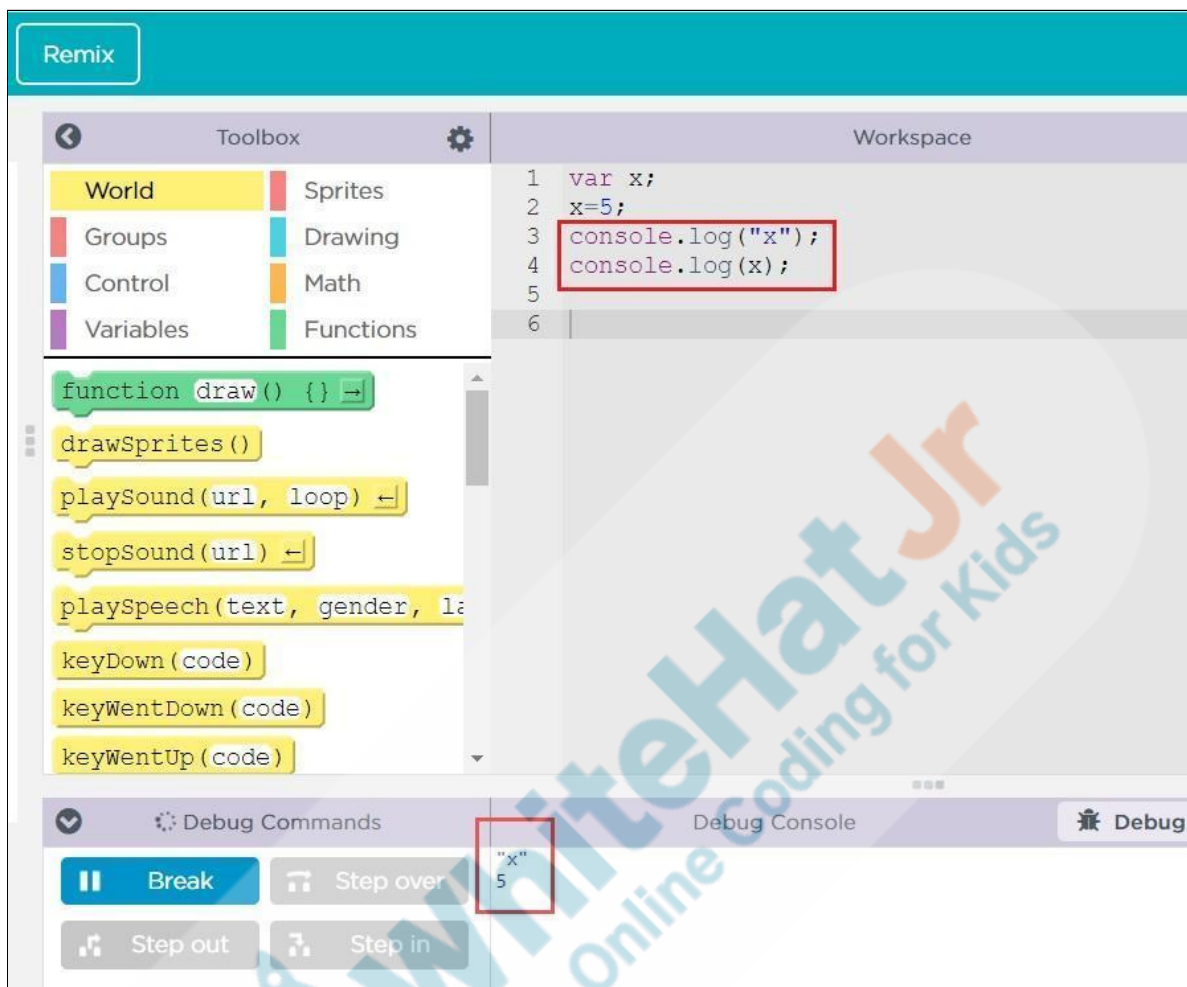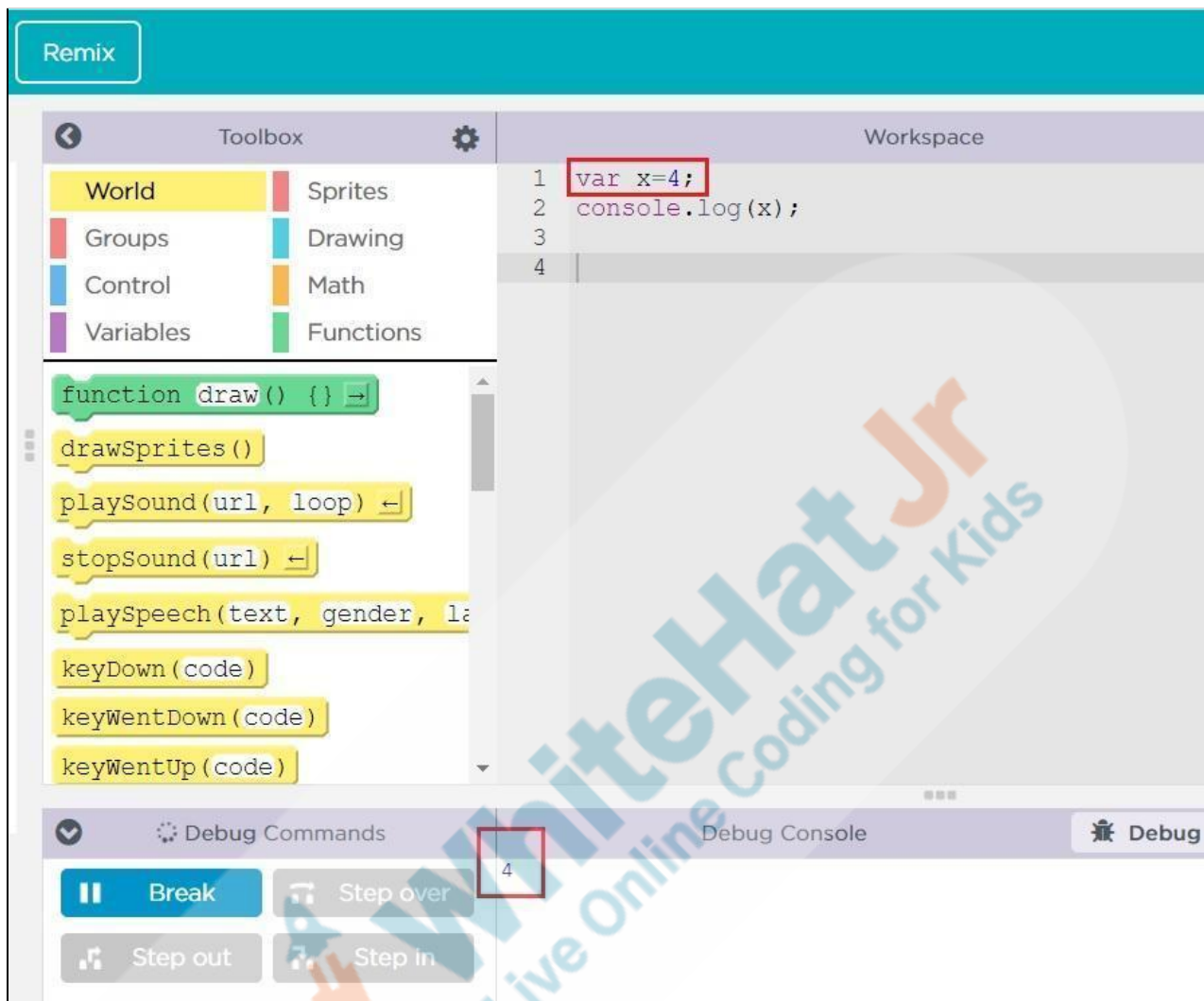|	ESR: Yes.
Did you notice we did not enclose '**x**' inside **double-quotes** as we did for "**Hello World**"?	
Can you tell me why?	ESR: Varied.
When we want to print a 'word' or a 'sentence' we use double-quotes but if we want to print the value of a variable, we write it without double-quotes in **console.log()**.	
Can you guess what will be the output of these two statements: **console.log("x");** **console.log(x);**	ESR: "x" and 5.

When we type, **console.log(x)**, the value of variable **x** appears on the screen (i.e. **5**) whereas **console.log("x")** prints **"x"** itself.

Hence, we can conclude that whatever we write in double quotes will be printed as it is.
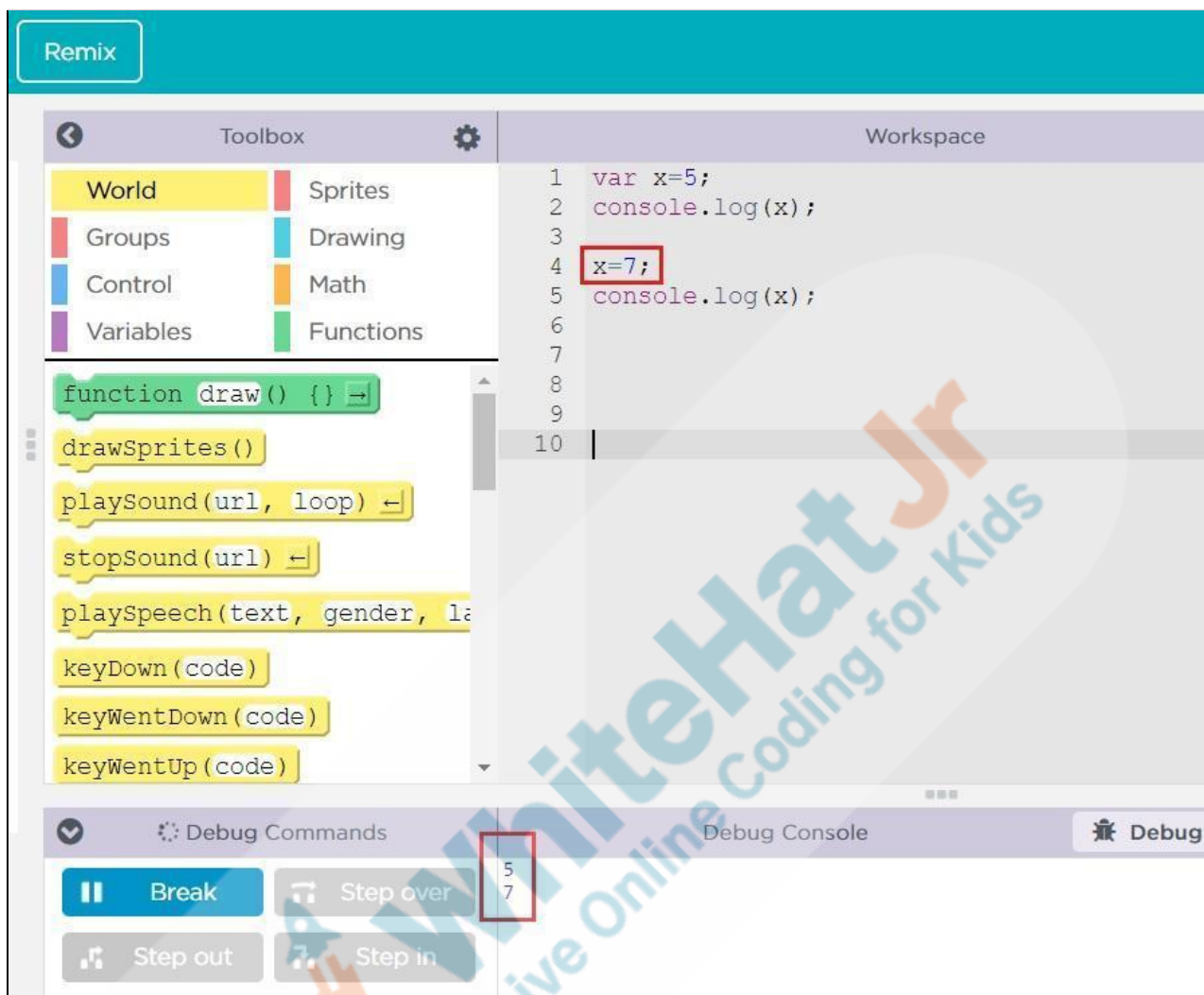
We can also assign an initial value to variables while declaring them. This is called **initialization**.

Note: If you use a variable without assigning it a value, it will have an **undefined value**.

We can also change the value of the variable anywhere in the code by simply using the assignment operator.

The teacher writes the code to change the value of the variable in the program and prints it.

Great! Now you know how to store different types of values inside variables and change them in the program.

You were fabulous today. It's your turn now.

Can you now write a program to create a variable to store your **current age/favorite number** and print it on the console?

ESR: Yes

Teacher Stops Screen Share

STUDENT-LED ACTIVITY 1 - 5 mins

	Now, it's your turn. Please share your screen with me.	

- **Ask the student to press the ESC key to come back to the panel.**
- **Guide the student to start screen share.**
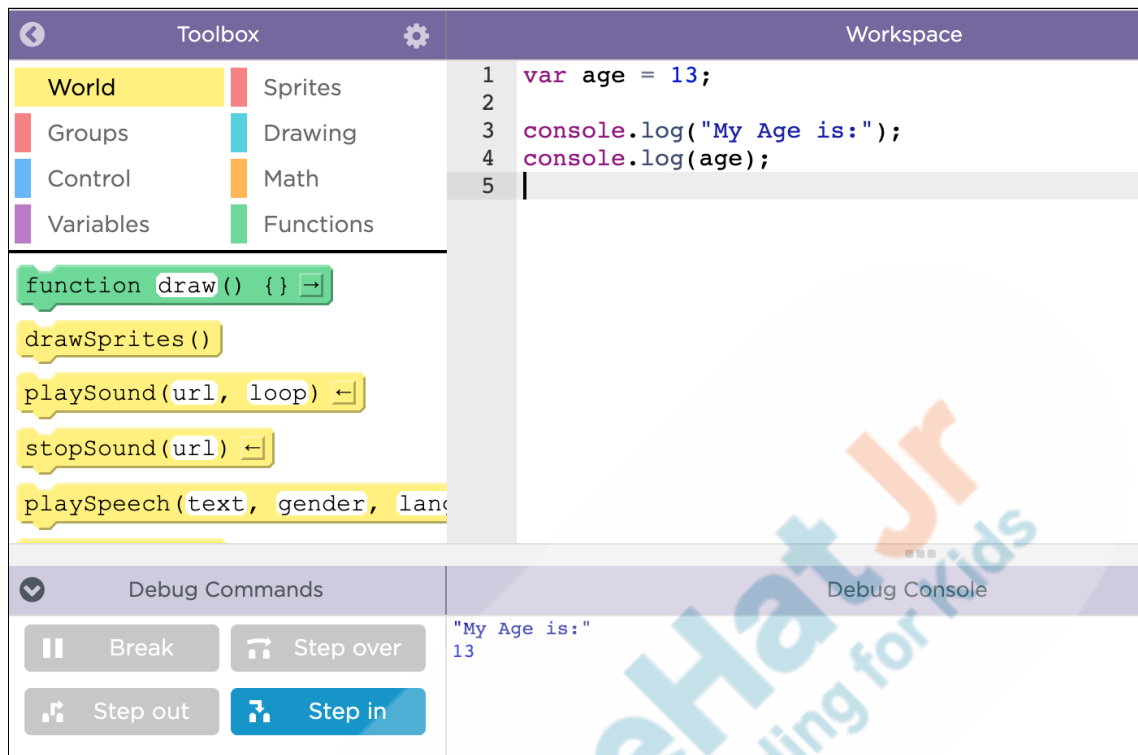- **Enter the Fullscreen mode.**

CHALLENGE
- **Student declares variables, assigns values to them, and prints them on the console.**

Teacher starts slideshow **for slide 11**
Refer to the speaker notes and follow the instructions on each slide.

Teacher Action	Student Action
Help the student open Student Activity 1 link and log in to code.org if he/she has not logged in. 1) Create a variable (preferably named '**age**') and initialize it with the actual age of the child. 2) Print the following on the console. *console.log("My Age is:");* *console.log(age);*	*The student clicks on Student Activity link 1 and starts coding to multiply two numbers and display the result on the console.*

Toolbox		Workspace
World	Sprites	
Groups	Drawing	
Control	Math	
Variables	Functions	

```
1  var age = 13;
2
3  console.log("My Age is:");
4  console.log(age);
5  |
```

Toolbox blocks:
```
function draw() {} →
drawSprites()
playSound(url, loop) ←
stopSound(url) ←
playSpeech(text, gender, lan
```

Debug Commands

- ❚❚ Break
- ⤼ Step over
- ⤶ Step out
- ⤷ Step in

Debug Console
```
"My Age is:"
13
```

Awesome. You did a great job. You have been understanding things very quickly. I am really impressed.

Now, let us learn about an interesting part of today's lesson, which is, Sprites. Sprites will help us create characters in our game. I will show you how to create a ghost sprite on my screen.

Are you excited?

Let me share my screen with you.

Teacher Guides Student to Stop Screen Share

Teacher can show slideshow **from slides 12 to 18**
Refer to the speaker notes and follow the instructions on each slide.

TEACHER LED ACTIVITY 2 - 10 mins

CHALLENGE

- **Creating and drawing the sprite.**

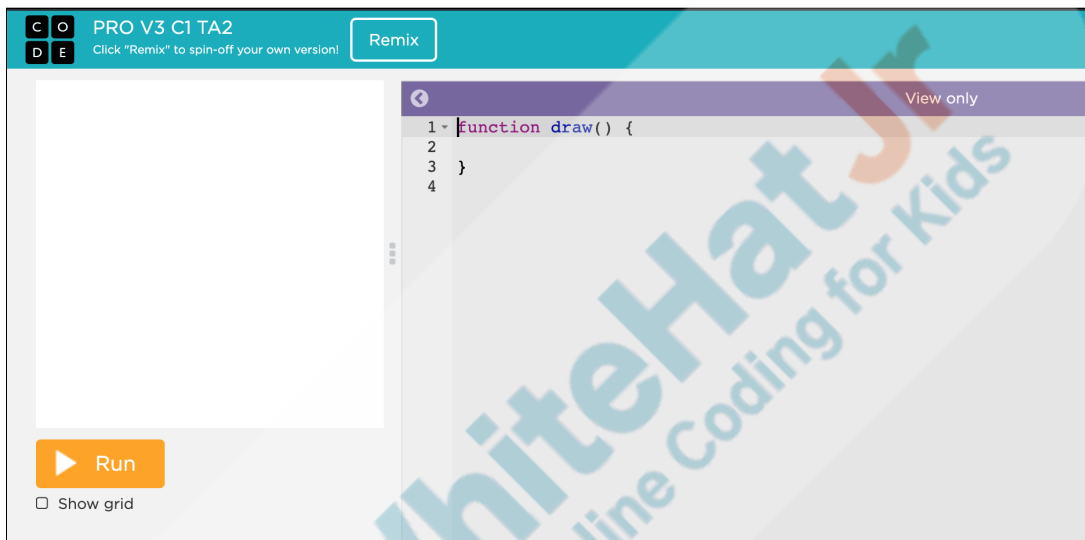| ● Change the sprite name, shapeColor property, and set animation for the sprite. |
|---|

Teacher-Led Activity (10 mins)	
Teacher Action	**Student Action**
The teacher clicks Teacher Activity 2 *and clicks* **View Code** *to view the code. The teacher then presses* **Remix** *and starts editing the code in* **Text mode***.*	Student watches.

Creating a game in **Game Lab** is like creating a **flipbook** where you decide your character first and then draw it repeatedly at different positions on different pages.

To see the output of the flipbook you have to run through all the pages at one go. This is done using the **draw()** function in **Game Lab**. Commands inside the **draw()** runs from top to bottom and once it reaches the end, it restarts from the top again. That is why it is also called a **draw loop**.

Whenever you create your game:

The things you want to set up once, come outside the **draw()** function, like creating a sprite or adding color to the sprite. Remember, **each page** on a flipbook is like one **frame** of the canvas which gets redrawn with each pass of the **draw()**.
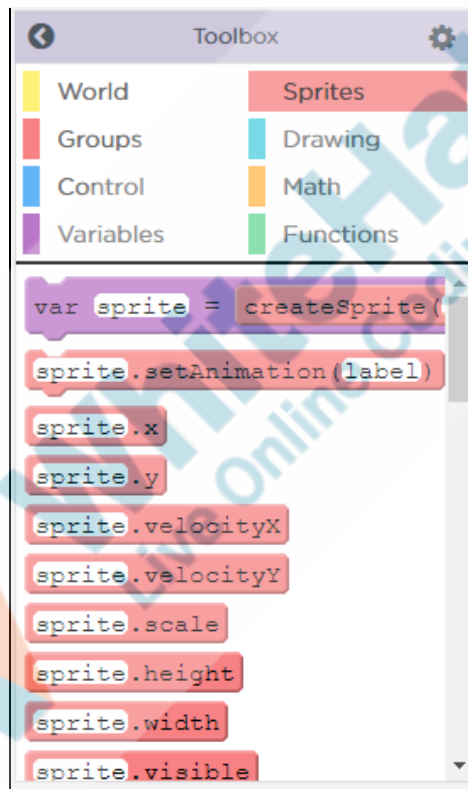
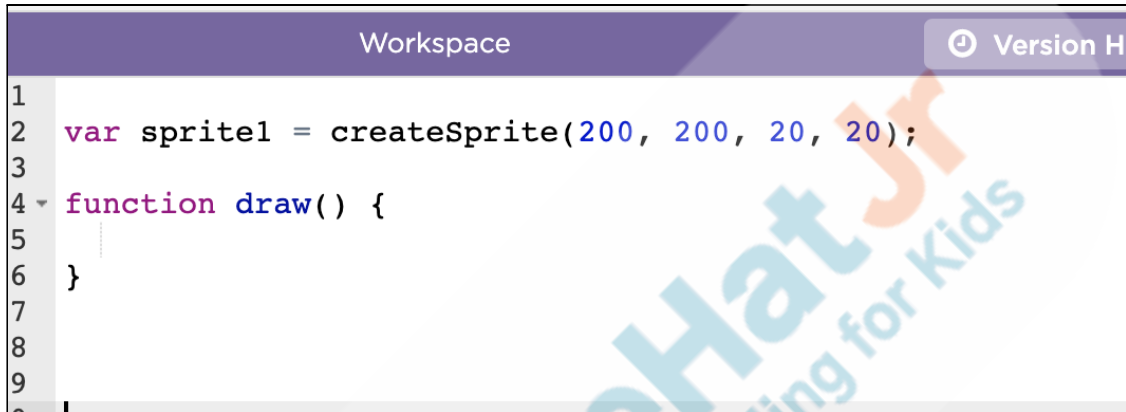|	
*Remember, the **draw()** function keeps running as long as the game is on.*	
Let us create a sprite now. A **Sprite** represents a character in a story, animation or game. It can be stored in a variable but unlike other variables **Sprite** is an object which can store multiple values under itself. Look into the **'Sprites'** section under our **Toolbox**. It should tell us how to create a **Sprite** object and what properties and functions it has.	The student looks into the **Sprite** section under **Toolbox**.
Do you see something which will help us create a **Sprite** object? To create a Sprite, we will have to tell the computer where we want to create the sprite on the canvas i.e. **x-position** and **y-position**. *The teacher clicks on the 'Show grid' (below the canvas) and shows x, y values of the mouse pointer to the student.* Also, we should tell the computer what will be the size of the Sprite	**ESR:** Yes! **var sprite = createSprite();**

i.e. **width** and **height**.

The teacher writes code for creating the sprite and runs the code.

The teacher asks the student for the x, y positions, width, and height of the ball.

CODE:

```
Workspace                                    Version Hi
1
2   var sprite1 = createSprite(200, 200, 20, 20);
3
4 ▾ function draw() {
5
6   }
7
8
9
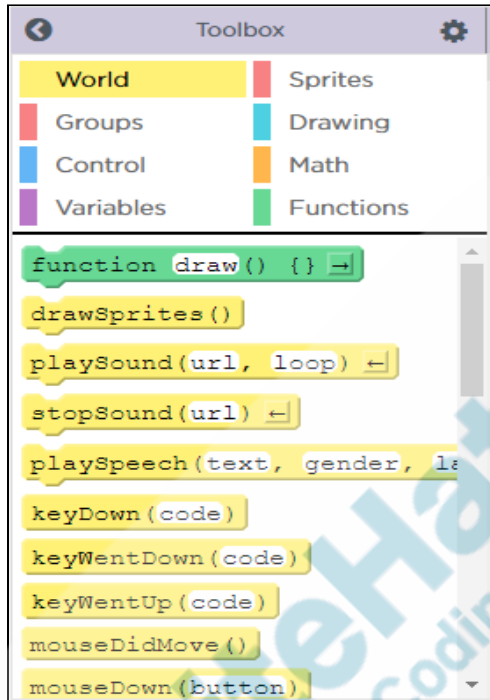```

**Output**:

```
Code  Animation
```

The screen doesn't show anything. What do you think happened?

I think the computer doesn't know that it has to draw the sprite on the screen. Let us see if there is an instruction to draw the sprite on the screen. Can you find one?

ESR: The student takes a guess at why the sprite hasn't appeared on the screen.

*The teacher scrolls the toolbox under the **World** section to locate the **drawSprites()** function.*	**ESR**: We will use the **drawSprites()** function.

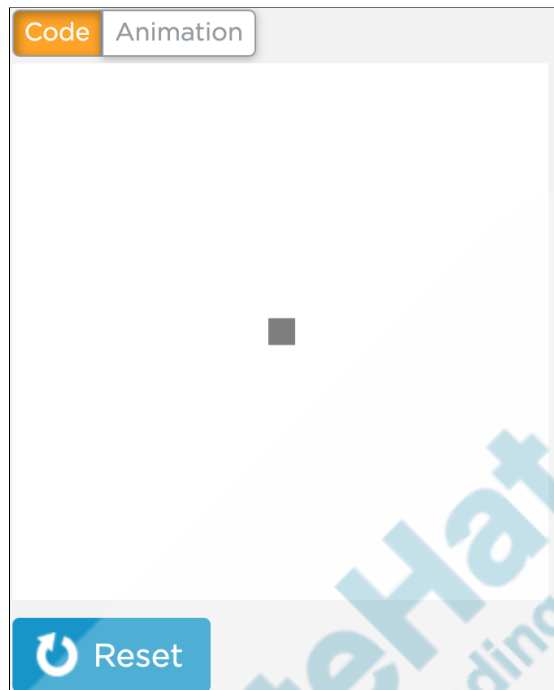|	
Since we want to draw the sprite in each and every frame we should write **drawSprites()** inside the **draw()** function. Let us write this code inside draw() and see what happens. *The teacher writes the code for drawing sprites and runs the code.*	The Student observes and learns.

CODE:

```
1
2  var sprite1 = createSprite(200, 200, 20, 20);
3
4  function draw() {
5      drawSprites();
6  }
7
```

OUTPUT:

Code | Animation

Reset

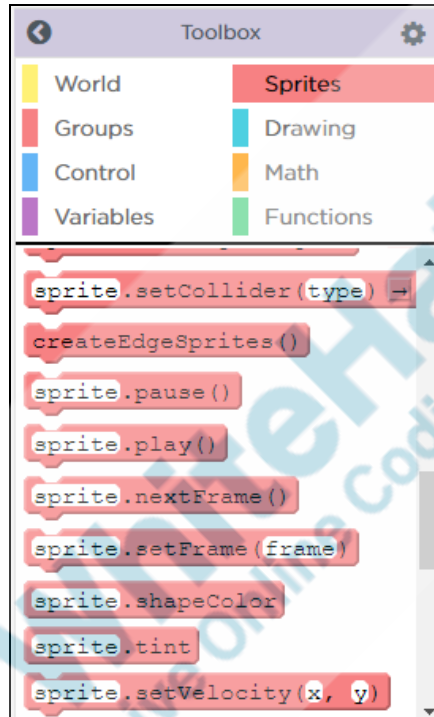|	
Can you see the sprite has been added to the output screen? **drawSprites()** draws all the sprites created in the code.	**ESR**: Yes!
Now let's try to change some properties of the sprite. *Click on the 'Sprite' tab in the Toolbox and show all the properties and methods available for Sprite objects.* All the properties and methods of the object can be accessed by using the **dot '.' operator**. Object name followed by the dot (**.**) operator and then the property name e.g. '**Sprite.width**'. *Properties are like predefined variables, and we can assign different values to them such as height, width, shapeColor, etc.* *Methods are predefined functions that perform a specific task such as setAnimation(), bounceOff(), etc*	**ESR**: methods are succeeded by **()**.

But, how do we differentiate between properties and methods in the list below for sprite objects? Functions/Methods always have **parentheses** at the end. E.g. **sprite.play()** is a sprite object method. Whereas, **sprite.shapeColor** is a sprite object property. (check Sprite object properties and methods in the Toolbox)	

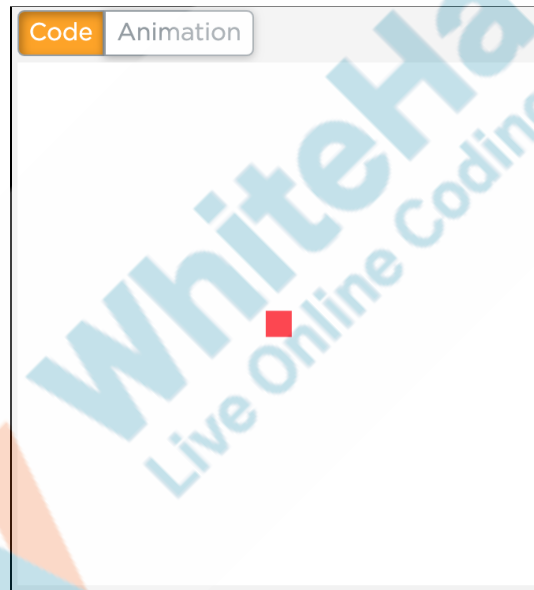|	
Let's first try changing some properties of the sprite object. What about the color of the sprite? Do you see any property related to sprite color in the list? We can change the property of a sprite by using its **object name**, **dot operator**, and **property name**. *The teacher assigns a 'red' value to the shapeColor property (variable) of the sprite and runs the code.*	**ESR**: sprite.shapeColor *The student observes and learns.*
CODE:	

```
                    Workspace                    ⏱ Version Histo
1   var sprite1 = createSprite(200, 200, 20, 20);
2   sprite1.shapeColor = "red";
3
4
5▾  function draw() {
6     drawSprites();
7
8   }
```

OUTPUT:

| Did you notice the color of the sprite change to red?

The teacher shows a few more options for color value. {Green, Blue, Pink, Yellow, etc}	**ESR**: Yes!
Great! Now, we know how to change the color of the sprite as per our preference.
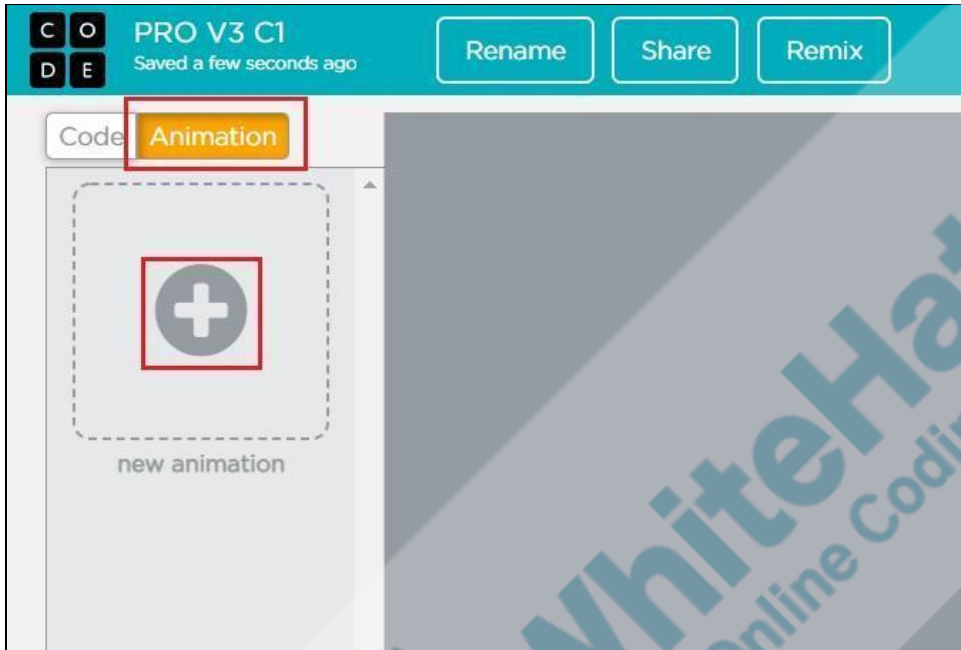
Did you notice the shape of the Sprite? | **ESR:** Rectangle. |

Correct. Sprites are always created rectangular. Wouldn't it be fun to have characters and animations as sprites?
Let me show you how to add animations or characters to our sprite.

The teacher will show the student how to add animation from the Animation tab.

ESR: Yes.

The student observes the code changes and the output.

Search for ghost_standing_1 animation from the list by clicking on new animation.

The teacher comes back to the Code tab.

Do you see any function to set the animation to a sprite?

The teacher drags the setAnimation() command from the sprite list and changes the sprite name and selects the animation name from the drop-down list of labels.

Run the code and see the ghost moving around in the canvas.

ESR: **setAnimation()**.

CODE:

Toolbox		Workspace

World	Sprites
Groups	Drawing
Control	Math
Variables	Functions

```
var sprite = createSprite(x,
sprite.setAnimation(label)
sprite.x
sprite.y
```

```
1  var sprite = createSprite(100, 100, 20, 20);
2
3  sprite.setAnimation("ghost_standing_1");
4
5
6  function draw() {
7    drawSprites();
8
9  }
10
11
12
13
14
15
```

OUTPUT:

Code   Animation

Reset

We now know how to create a sprite and change its properties and animation.

It's your turn now to create a ball sprite and play with its properties. I will help you with the same. I have also left comments for your help in the program. Feel free to ask me any questions you have.

Ready?

ESR: Yes!

Teacher Stops Screen Share

STUDENT-LED ACTIVITY 2 - 10 mins

- **Ask Student to press ESC key to come back to the panel**
- **Guide Student to start Screen Share**
- **Teacher gets into Fullscreen**

CHALLENGE

- **Create a Ball Sprite and play with its properties (shapeColor, scale)**
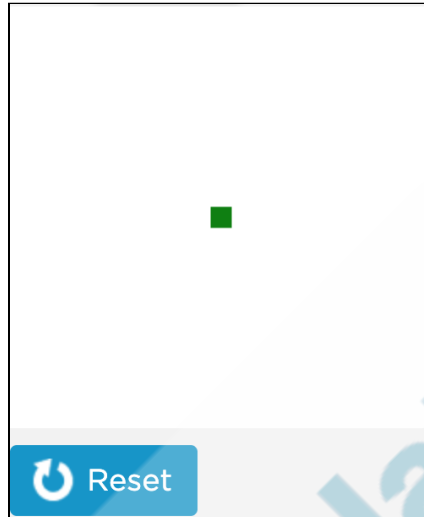- **Set animation to the sprite.**

Teacher can show slideshow from slides **19 & 20**
Refer to speaker notes and follow the instructions on each slide.

Teacher Action	Student Action
Guide the student to create a ball object. Begin by creating a new **Sprite** object at center(200,200); name it **ball**. Change the color of the ball as per preferred color by the student. *Observe the student, check for any typos she/he makes and guide the student to write the code. Please note the color name should be in double quotes as it's a constant string.*	*The student opens Student Activity 2 and presses 'View Code'. Click on 'Remix'.* *The student writes code to create a new sprite.*

CODE:

```
                          Workspace
1    var ball;
2
3    ball = createSprite(200,200,20,20);
4    ball.shapeColor ="green";
5
6 ▾  function draw() {
7
8       drawSprites();
9
10   }
11
12
```

OUTPUT:

Reset

Now add the ball animation to the sprite.

We no longer need to assign color to the sprite so you can remove **shapeColor** instruction from the program.

Observe the student, check for any typos she/he makes. Guide the student to write the code.

The student writes code to assign the ball animation to the sprite.

CODE:

```
Workspace
1   var ball;
2
3   ball = createSprite(200,200,20,20);
4   ball.setAnimation("volleyball2_1");
5
6   function draw() {
7
8       drawSprites();
9
10  }
11
12
```

OUTPUT:

But the object here is too big to fit the screen size. Right?

Don't you think we will need to resize the ball object to a smaller size?

ESR: Yes.

Let's check under the **Sprite** tab, what instructions can be used for resizing the sprite. We can use the **scale** property.

Scale helps us shrink or expand the size of sprite. Value of scale should be always positive.
For e.g. 2 will make the size twice as big and 0.5 will make it half the size. By default the value of scale is 1 which shows the original size of the sprite.

Can you try different values to scale the Ball Sprite to an appropriate size?

Observe the student, check for any typos she/he makes. Guide the student to write the code. (Note: scale value of 0.1 would be ideal for our game)

ESR: Yes

The student writes code to scale the ball object.

He/She runs the code to see the output.

CODE:

```
                           Workspace
 1   var ball;
 2
 3   ball = createSprite(200,200,20,20);
 4   ball.setAnimation("volleyball2_1");
 5   ball.scale = 0.1;
 6
 7 ▾ function draw() {
 8
 9      drawSprites();
10
11   }
12
13
```

OUTPUT:

This looks amazing! You seem to have done pretty well in this challenge.

Now that you are a programmer, let me tell you a good practice programmers follow:
A good programmer always writes code in a way that is easy to understand for everyone. Even your friend should be able to read the code and understand the code you have written. We use **comments** for it.

Comments are added using the '**//**' before a line.
Let us quickly add more comments wherever required in our code.

```
                        Workspace
 1    var ball;
 2
 3    //create ball sprite
 4    ball = createSprite(200,200,20,20);
 5
 6    //set animation to volleyball
 7    ball.setAnimation("volleyball2_1");
 8
 9 ▾  function draw() {
10      //draw all the sprites on the canvas
11      drawSprites();
12
13    }
14
15
16
```

Great, you were amazing throughout the class!
Remember, Comments are helpful not only for others but also for the programmer to understand what each line of the code stands for, at any time they look at a code again.

Teacher Guides Student to Stop Screen Share

WRAP-UP SESSION - 5 mins

FEEDBACK
- **Appreciate and compliment the student for trying to learn a difficult concept.**
- **Get to know how they are feeling after the session.**
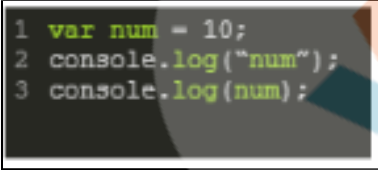- **Review and check their understanding.**

Teacher can show slideshow ⬜ **from slides 21 to 31**
Refer to the speaker notes and follow the instructions on each slide.

For the 'Wrap-up' section, there will be slides on the panel as visual aid to summarize what has been done in the class.

Step 4: Wrap-Up (5 min)	There are a few things that we learned today which are new, and we will take some time to get used to it - which is fine. For now, can we quickly summarize what we have learned today?	**ESR:** - We created sprites and changed its color. - We added animation and character to the ball sprite. - We scaled the ball object.

<table>
<tr><td colspan="2" align="center">QnA Session - Click on in-class quiz</td></tr>
<tr><td align="center">Question</td><td>Answer</td></tr>
<tr><td>Which keyword is used in JavaScript to declare a variable?

  A. var
  B. variable.
  C. Int
  D. None of the above</td><td>A</td></tr>
<tr><td>What will be the output of the following code?


```
1 var num = 10;
2 console.log("num");
3 console.log(num);
```

    A. error

    B. num
       undefined.

    C. 10
       num

    D. num
       10</td><td>D</td></tr>
</table>

Which function is used to write messages to the consoles?
 	
  A. inspect.log()
  B. console.log()
  C. gamelab.log()
  D. log.console()
	B

<div align="center">**End the quiz panel**</div>

You get Hats off for your amazing performance today. Alright, we seemed to have a lot of learning in the class today. Today's class was slightly difficult, but you did extremely well. In the next class, we will make our sprites move, and we will also learn how to make the ball bounce from the edges! Isn't that interesting!	*Make sure you have given at least 2 Hats Off during the class for:* Creatively Solved Activities +10 Great Question +10 Strong Concentration +10
	Project Overview **ROCKETS: Ready to Launch** **Goal of the Project:** In Class 1, you learned about the concept of variables and objects. You learned how to create sprite objects, alter their properties, and set an animation to the sprite. You will write code to create a rocket lift-off animation. In this project, you will have to practice and apply what you have learned in the class and create a scene of rockets ready to launch. **Story:** Daisy loves to read about Rockets and Space, where her father helps her with astronomical	*The student engages with the teacher over the project.*

events and rocket launches. Daisy has just embarked on a coding learning journey, and she is eager to try her hand at creating a rocket sprite and show it to her father.

Help Daisy build such a scene where two rockets are getting ready to fly into the sky and vanish into space.
I am very excited to see your project solution and I know you will do really well.

Bye Bye!

Teacher ends slideshow

ADDITIONAL ACTIVITIES (For fast students)
Ask student to share the screen and continue on the same code

Remember you created the first sprite at (200, 200) and of size 20 x 20.

Here is the challenge for you to create 3 more sprites in the same program but with some change in either the position(x,y) or width of the sprite or height of the sprite such that we get to see four different sprites on the screen.

*The teacher guides the student in creating 3 more sprites with different names and changes the **x, y values** in the 2nd sprite, **width** in the 3rd sprite, **height** in the 4th sprite.*

Run the code to see the four sprites.

CODE:

```
Workspace                          ⏱ Version History
var sprite1 = createSprite(200, 200, 20, 20);
var sprite2 = createSprite(200, 150, 20, 20);
var sprite3 = createSprite(150, 100, 30, 20);
var sprite4 = createSprite(100, 200, 20, 40);

function draw() {
  drawSprites();

}
```

OUTPUT:

Can you tell me which one of the four is sprite3?	ESR: Varied.
The teacher asks the child to find out which one is the sprite 4 and a few more. The teacher also encourages the child to check the position of the sprite using 'Show Grid'.	
Do you know, we have one more way to find out the sprites without doing too much of these analysis.	

We can simply assign different colors to different sprites by using the *'shapeColor'* property.

*Encourage the child to add **shapeColor** property for each sprite in the code and test it again.*

Additional Activity 2: Javascript Data Types.

JavaScript Data Types: JavaScript's variables are used to store different types of data like numbers, decimals, and text. In programming, text values are called **Strings**.
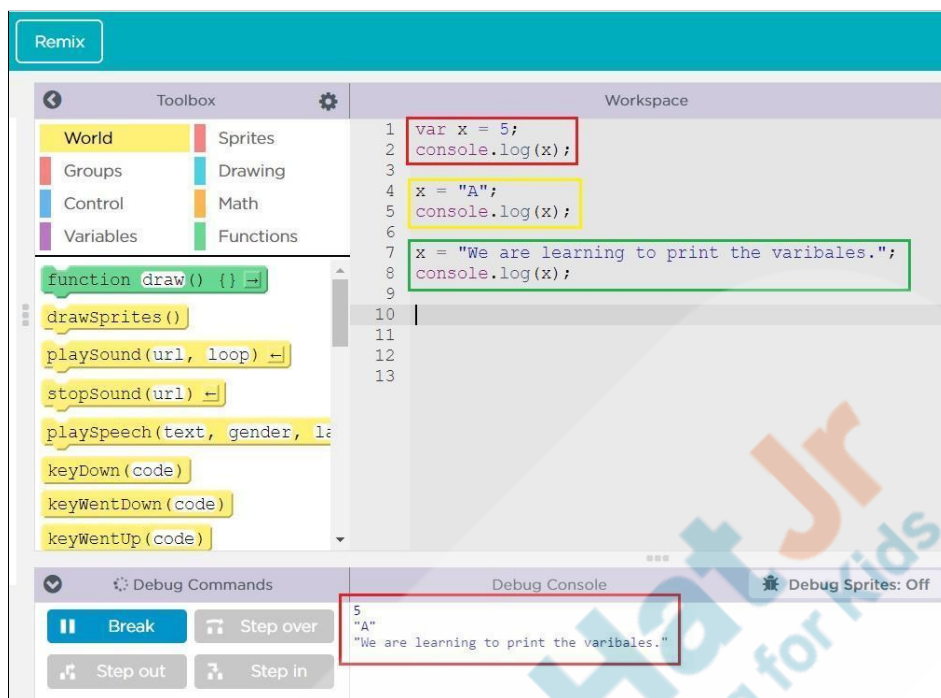
JavaScript can handle many types of data, but for now, we will talk about numbers, strings, and objects.

var planets = 8; //Numbers are written without quotes.
var name = "Harry Potter"; // Strings are written inside double or single quotes. If you put a number in quotes, it will be treated as a string.
var car = {color: "Black", Engine: "1.2L"}; //Objects contain many values. We will learn more about objects in the later half of class.

Teacher to store different types of values and print them in the program for the student to observe.

Encourage the child to store his/her name, age and address in three different variables. Finally print all three details on the console.

Activity	Activity Name	Links
Teacher Activity Link 1	Intro to Variables	https://studio.code.org/projects/gamelab/5KmzdhP1NsX5h7fhPS4IL_VbFddRr69S_g9wBOBaPW0
Teacher Activity Link 1 (Ref Code)	About Variables (reference code)	https://studio.code.org/projects/gamelab/A51SdXPY0_CWEsg5XU4uy1jlZ4c_9UmVrxIHC3hm9jM
Student Activity Link 1	Variable Activity	https://studio.code.org/projects/gamelab/aaLbW9OTrptjRCwuPF2JVM79ZnYfv5SftUKO_Zt0Dyc
Student Activity Link 1(Ref Code)	Variable Activity (reference code)	https://studio.code.org/projects/gamelab/Yo0UM2kETSroyWehl8cBydLX7d0Xop0YwOVPzuTGSl8
Teacher Activity Link 2	Ghost Animation	https://studio.code.org/projects/gamelab/q00kjjDQWyg-Rs0gVubYpPkTj7M3WPCiN5c0fBG9WPs

Teacher Activity Link 2 (Ref Code)	Ghost animation (reference code)	https://studio.code.org/projects/gamelab/nWCH32sO9bP6Du4hg7u1UTZzF3ydj6Mlz5YpxfYRvVY
Student Activity Link 2	Ball animation	https://studio.code.org/projects/gamelab/K1ZZ1r2DNBIv1JR684Y4zGQEtQdmsGjUQJMbfx4uclo
Student Activity Link 2 (Ref Code)	Ball animation	https://studio.code.org/projects/gamelab/Rael5YgIHRh6Xf962gYZZSTfT7EpaRmrCAdT6S7Sj7o
Additional Activity 2 (Ref Code)	JavaScript Data Types	https://studio.code.org/projects/gamelab/2TXy_FrKKDBsbfB-weI9F5l9O6VnKuFoTWesN74KfsQ
Additional Activity 1 (Ref Code)	Play with Sprite properties (Position and size)	https://studio.code.org/projects/gamelab/2TXy_FrKKDBsbfB-weI9Fwk8rJA-YEgP6Gor7CHXq64
Teacher Reference visual aid link	Visual aid link	https://curriculum.whitehatjr.com/Visual+Project+Asset/PRO_VD/BJFC-PRO-V3-C1L-withcues.html
Teacher Reference In-class quiz	In-class quiz	https://s3-whjr-curriculum-uploads.whjr.online/00df23ba-16bc-4079-a9ce-e8ff49e8e88c.pdf