c1:

**console.log:**

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

Steps on 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.

**Review the “Game Lab” in code.org.**

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.

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

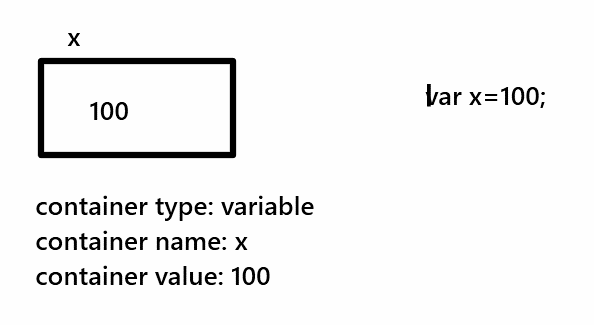
**"Text mode"** by clicking **“Show Text”**.

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.

**Variables**

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.

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

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.

**draw()**

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.

**Sprites**

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**

Also, we should tell the computer what will be the **size** of the Sprite i.e. **width and height**

Sprites are always created rectangular.

Since we want to draw the sprite in each and every frame we should write **drawSprites**() inside the draw() function.

drawSprites() draws all the sprites created in the code.

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

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**.

We can **change** the property of a sprite by using its object name, dot operator, and property name.

**Scale:**

For resizing the sprite, we can use the scale property

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.

Eg: sprite.scale=0.5

**Comments:**

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