SCOPE OF VARIABLES

What is our GOAL for this MODULE?

Use our knowledge of variables, functions, loops, game states, etc., to reset the game and set up a local environment to run the Trex code locally.

What did we ACHIEVE in the class TODAY?

- Changed the scope of the variables from local to global for the **Reset** and **GameOver** sprites
- Reset the game when the reset icon is pressed

Which CONCEPTS/ CODING BLOCKS did we cover today?

- Scope of variables
- Changing the game state

How did we DO the activities?

1. Experiment with the scope of the variable at the correct place to declare them as per their role in the game. When the variables are used out of scope, it gives a reference error. Thus, variables have a scope and they live and die inside the scope.

```
var PLAY = 1;
var END = 0;
var gameState = PLAY;

var trex, trex_running, trex_collided;
var ground, invisibleGround, groundImage;

var cloudsGroup, cloudImage;
var obstaclesGroup, obstacle1, obstacle2, obstacle3,
obstacle4, obstacle5, obstacle6;

var score;
var gameOverImg,restartImg
var jumpSound , checkPointSound, dieSound
```

2. Add a reset icon and add functionality to it. Use the **mousePressedOver()** instruction to detect if the mouse is pressed over the reset sprite and print a message when it is pressed:

```
    obstaclesGroup.setVelocityXEach(0);
    cloudsGroup.setVelocityXEach(0);
  }

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

  if(mousePressedOver(restart)) {
      console.log("Restart the Game")
    }


  drawSprites();
}
```

3. Create a **reset()** function which resets everything in the game to its original state.

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

  if(mousePressedOver(restart)) {
      console.log("Restart the Game")
    }


  drawSprites();
}

function reset(){

}
```

4. Write code for the **reset()** function:

```
function reset(){
  gameState = PLAY;
  gameOver.visible = false;
  restart.visible = false;
}


function spawnObstacles(){
 if (frameCount % 60 === 0){
   var obstacle = createSprite(600,165,10,40);
   obstacle.velocityX = -(6 + 3*score/100);
```

5. Destroy the **obstacles** when the game is reset.

```
function reset(){
  gameState = PLAY;
  gameOver.visible = false;
  restart.visible = false;

  obstaclesGroup.destroyEach();
  cloudsGroup.destroyEach();
}


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

6. Change the **Trex** collided animation to Trex running.

```
function reset(){
  gameState = PLAY;
  gameOver.visible = false;
  restart.visible = false;

  obstaclesGroup.destroyEach();
  cloudsGroup.destroyEach();

  trex.changeAnimation("running", trex_running);
}
```

7. Move the condition where we checked the mouse pressed over reset icon inside game end state to fix the restarting game issue even when the reset icon was invisible. Reset the **score.**

```
  drawSprites();
}

function reset(){
  gameState = PLAY;
  gameOver.visible = false;
  restart.visible = false;
  trex.changeAnimation("running", trex_running);

  obstaclesGroup.destroyEach();
  cloudsGroup.destroyEach();
  score = 0;

}
```

8. Update the **getFrameRate()** function that gives the number of frames displayed in the game per second.

```
if(gameState === PLAY){
  //move the
  gameOver.visible = false;
  restart.visible = false;

  ground.velocityX = -(4 + 3* score/100)
  //scoring
  score = score + Math.round(getFrameRate()/60);

  if(score>0 && score%100 === 0){
     checkPointSound.play()
  }

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


What's next?

Learn to write code on the local machine. We will also modify our Trex code slightly so that our game can be run on a mobile phone.

Extend Your Learning:

Learn more about the scope of variables.