**Assignment 1:**

**Name: David Andres Barrios Rodriguez**

**Student ID: C0893262**

**Exercise 1:** Analyzing Variable Hoisting

\*\*Objective:\*\* Identify the behavior of variable hoisting in the provided code snippets.

\*\*Instructions:\*\*

- Review each code snippet.

- Predict the output.

- Run the code to check your predictions.

\*\*Code Snippets:\*\*

1. ```javascript

console.log(crewMember); // ? 1st

var crewMember = 'Alice';

console.log(crewMember); // ? 2nd

```

2. ```javascript

var mission = 'Mars Exploration';

console.log(mission); // ? 3rd

var mission = 'Lunar Landing';

console.log(mission); // ? 4th

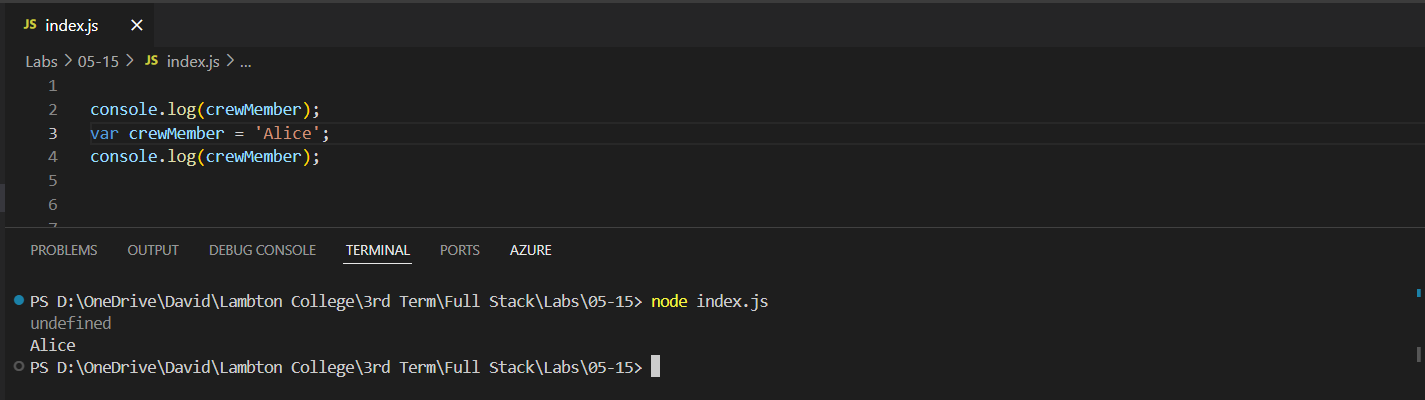
```

\*\*Questions:\*\*

1. **What is the output of each `console.log` statement?**

For the 1st console.log the output is *undefined,* for the 2nd one: “Alice”, for the 3rd one is “Mars Exploration” and for the 4th one is “Lunar Landing”.

1st and 2nd console.log:



3rd and 4th console.log:

A screenshot of a computer

Description automatically generated

1. **How does hoisting affect the variable declarations?**

By letting the user use them before declaring them, but as they are not initialized, in case they use them, they are undefined, but the program execution doesn’t throw any error.

**Exercise 2:** Avoiding Temporal Dead Zone Errors

\*\*Objective:\*\* Understand and avoid the Temporal Dead Zone errors.

\*\*Instructions:\*\*

- Examine each code snippet.

- Predict if the code will throw an error.

- Run the code to confirm your prediction.

\*\*Code Snippets:\*\*

1. ```javascript

console.log(spacecraft); // ? 1st

let spacecraft = 'Orion';

console.log(spacecraft); // ? 2st

```

2. ```javascript

const destination = 'Mars';

console.log(destination); // ? 3rd

{

console.log(destination); // ? 4th

let destination = 'Moon';

console.log(destination); // ? 5th

}

```

\*\*Questions:\*\*

1. **Which `console.log` statements will throw errors and why?**

1st one throws an error because as *spacecraft* isdeclared with let after being invoked, it falls on the Time dead zone and the program doesn’t know that this variable is declared afterwards, and 4th one because destination goes out of the scope after enclosing it in curly braces.

**1st console.log:**

**A screenshot of a computer

Description automatically generated**

**4th console.log error:**

A screenshot of a computer program

Description automatically generated

1. **How can you refactor the code to avoid TDZ errors?**

In the first one simply by changing the order of the declaration “let spacecraft = 'Orion';” before the first console.log declaration. And the fourth one by deleting the block and deleting the “let destination = ‘Moon’” line. Or simply by switching positions between console.log(destination); and let destination = ‘Moon’;.

**Fixing 1st console.log:**

A screenshot of a computer

Description automatically generated

**Fixing 4th console.log:**

A screenshot of a computer

Description automatically generated

**Exercise 3:** Refactoring Code for Better Hoisting Practices

\*\*Objective:\*\* Refactor code to improve readability and avoid potential hoisting pitfalls.

\*\*Instructions:\*\*

- Refactor the given code snippets to use best practices for variable declarations.

- Ensure the code runs without errors.

\*\*Code Snippets:\*\*

1. function launchRocket() {

console.log(rocketType);

var rocketType = 'Falcon Heavy';

console.log(rocketType);

}

launchRocket();

```

2. ```javascript

var launchTime = '10:00 AM';

function scheduleLaunch() {

console.log(launchTime);

var launchTime = '2:00 PM';

console.log(launchTime);

}

scheduleLaunch();

```

\*\*Task:\*\*

**Refactor the code so that variable declarations are clear and predictable.**

A screenshot of a computer

Description automatically generated

Code 1 was fixed just by eliminating the line “console.log(rocketType);” because the variable rocketType was not initialized yet.

A screenshot of a computer

Description automatically generated

Similarly, this one was fixed just by deleting the line containing the variable launchTime before it was initialized.

**Exercise 4:** Debugging Temporal Dead Zone Issues

\*\*Objective:\*\* Identify and fix TDZ issues in the provided code snippets.

\*\*Instructions:\*\*

- Analyze the code snippets for TDZ issues.

- Correct the code to prevent errors.

\*\*Code Snippets:\*\*

1. ```javascript

console.log(astronaut); // ?

let astronaut = 'Neil Armstrong';

```

2. ```javascript

{

console.log(oxygenLevel); // ?

const oxygenLevel = 'Safe';

console.log(oxygenLevel); // ?

}

```

\*\*Task:\*\*

**Identify the lines causing TDZ errors and modify the code to eliminate these errors.**

A screenshot of a computer

Description automatically generated

For this one is enough by declaring and initializing the variable astronaut before being called in ethe console.log line.

A screenshot of a computer

Description automatically generatedIn this case, we fix the TDZ by avoiding to use the variable before using it on console.log as shown in the image.

**Exercise 5:** Implementing Safe Variable Declarations

Objective: Implement best practices for variable declarations to avoid hoisting and TDZ issues.

\*\*Instructions:\*\*

- Rewrite the given code snippets using `let` and `const`.

- Ensure the code is clean and error-free.

\*\*Code Snippets:\*\*

1. ```javascript

var base = 'Alpha';

console.log(base);

var base = 'Bravo';

console.log(base);

```

2. ```javascript

function prepareRations() {

var rations = 'Ready';

console.log(rations);

var rations = 'Packed';

console.log(rations);

}

prepareRations();

\*\*Task:\*\*

**Use `let` and `const` to declare variables and refactor the code for better readability and error prevention.**

A screenshot of a computer

Description automatically generated

For this one we switch the first declaration to let, and then when we are changing its value, we need to omit the let, because the variable is already initialized.

A screenshot of a computer

Description automatically generated

Similarly, for this case we simply initialize the variable rations to *let* and when we are assigning the variable again we need to rid of the *var* keyword and the code will run without errors.

**Exercise 6:** Building a Hoisting-Safe Mars Rover Control System

\*\*Objective:\*\* Create a simple Mars rover control system that adheres to best practices for variable hoisting and TDZ.

Instructions:

- Design a control system with commands to move the rover.

- Ensure all variable declarations use `let` or `const`.

- Test the system to confirm it works correctly without hoisting or TDZ issues.

\*\*Sample Code Structure:\*\*

function MarsRoverControl() {

let position = { x: 0, y: 0 };

const commands = ['MOVE\_FORWARD', 'TURN\_LEFT', 'TURN\_RIGHT'];

function executeCommand(command) {

switch (command) {

case 'MOVE\_FORWARD':

position.y += 1;

break;

case 'TURN\_LEFT':

position.x -= 1;

break;

case 'TURN\_RIGHT':

position.x += 1;

break;

default:

console.log('Invalid command');

}

}

function getPosition() {

return position;

}

return {

executeCommand,

getPosition

};

}

const rover = MarsRoverControl();

rover.executeCommand('MOVE\_FORWARD');

console.log(rover.getPosition()); // { x: 0, y: 1 }

rover.executeCommand('TURN\_LEFT');

console.log(rover.getPosition()); // { x: -1, y: 1 }

```

**\*\*Task:\*\***

**Complete the control system implementation, ensuring all commands work correctly and variables are safely declared.**

Answer: A screenshot of a computer program

Description automatically generated

Code in text format:

function MarsRoversControl(){

    let positionX = 0;

    let positionY = 0;

    function move(command){

        switch (command) {

            case "1": // Go ahead

                positionY += 1;

                break;

            case "2": // Go right

                positionX +=1;

                break;

            case "3": // Go left

                positionX -=1;

                break;

            case "4": // Go back

                positionY -=1;

                break;

            default:

                break;

        }

    };

    function getPositionX() {

        return positionX;

    }

    function getPositionY() {

        return positionY;

    }

    return {move, getPositionX, getPositionY};

}

const rovers = MarsRoversControl();

let movement = "1";

rovers.move(movement);

console.log(`Position: x = ${rovers.getPositionX()}, y = ${rovers.getPositionY()}. Moving ahead`);

movement = "2";

rovers.move(movement);

console.log(`Position: x = ${rovers.getPositionX()}, y = ${rovers.getPositionY()}. Moving right`);

movement = "1";

rovers.move(movement);

console.log(`Position: x = ${rovers.getPositionX()}, y = ${rovers.getPositionY()} Moving ahead, again`);