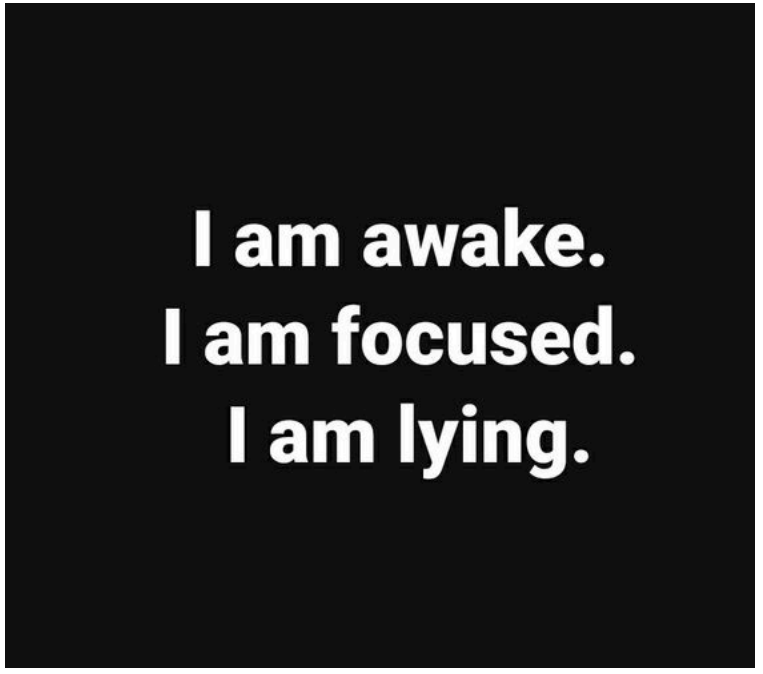
Codergirl - JavaScript

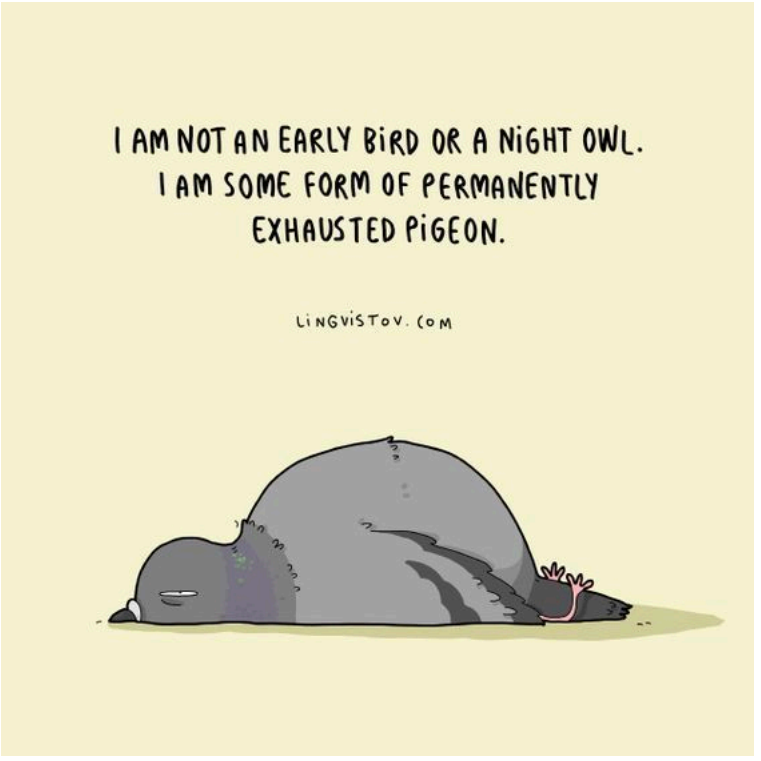
Class 10

April 28, 2021

Agenda

- Studio recap – end at 6:00 pm
- Lecture – end at 6:30 pm
- Exercise – end at 7pm





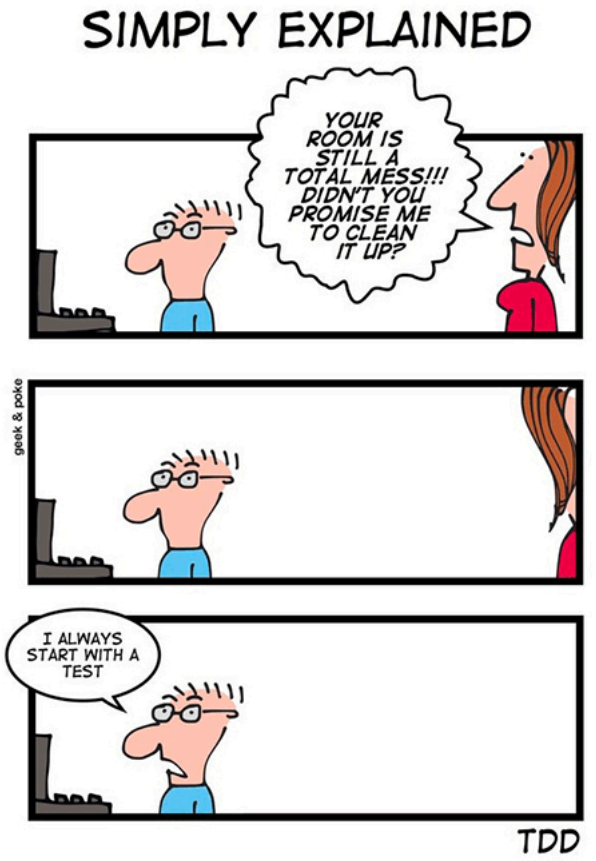
Studio Unit Testing

Rather than complete the code and *then* test it, TDD flips the process:

1. Write a test first - one that checks the program for a specific outcome.
2. Run the test to make sure it fails.
3. Write a code snippet that passes the test.
4. Repeat steps 1 - 3 for the remaining features of the program.
5. Examine the code and test and refactor to increase efficiency. Remember the DRY idea (Don't Repeat Yourself).

OR

RED – GREEN – REFACTOR



Studio Unit Testing

Let's use Test Driven Development (TDD) to help us design a function that meets the following conditions:

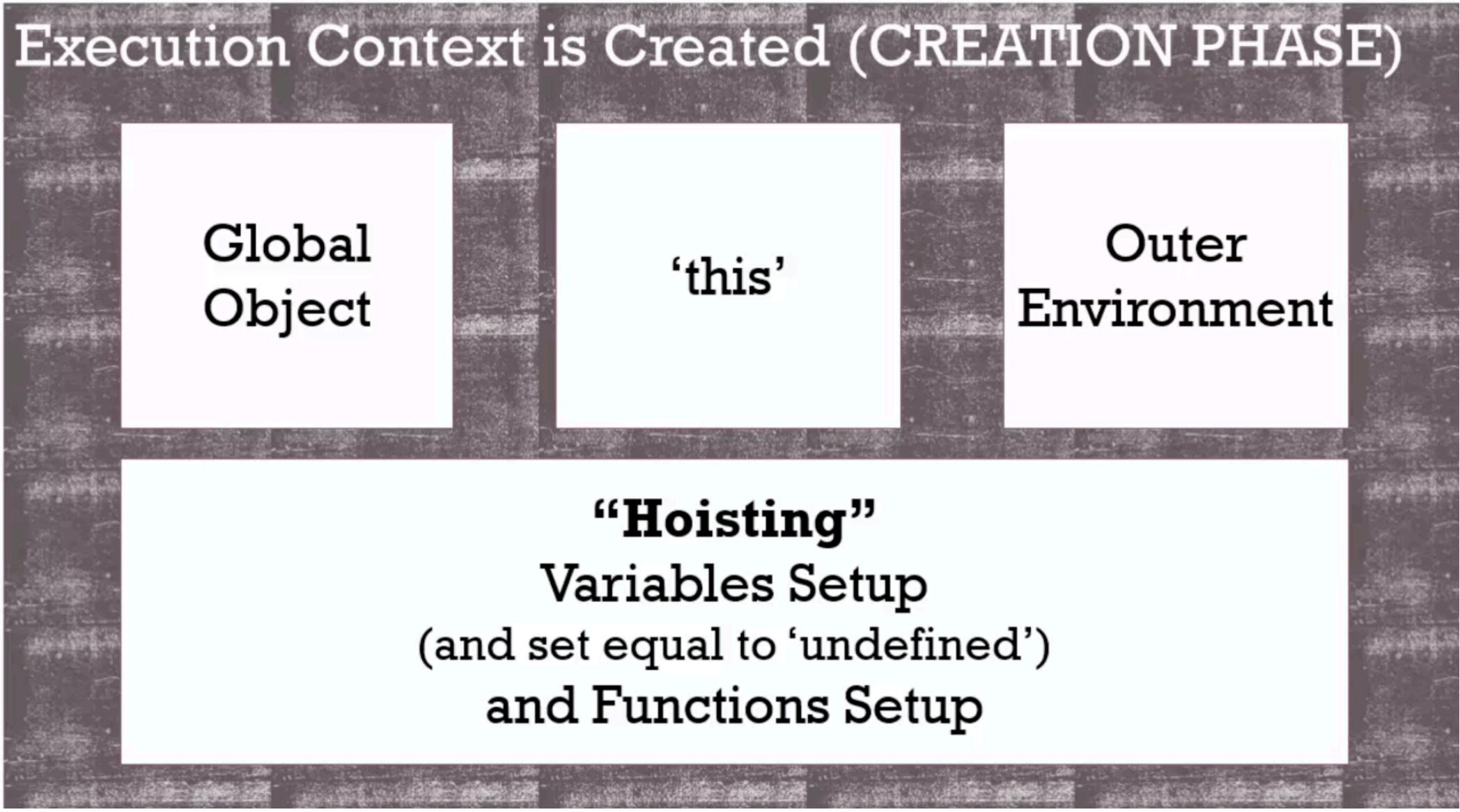
1. When passed a number that is ONLY divisible by 2, return 'Launch!'
2. When passed a number that is ONLY divisible by 3, return 'Code!'
3. When passed a number that is ONLY divisible by 5, return 'Rocks!'
4. When passed a number that is divisible by 2 AND 3, return 'LaunchCode!'
5. When passed a number that is divisible by 3 AND 5, return 'Code Rocks!'
6. When passed a number that is divisible by 2 AND 5, return 'Launch Rocks!'
7. When passed a number that is divisible by 2, 3, AND 5, return 'LaunchCode Rocks!'
8. When passed a number that is NOT divisible by 2, 3, or 5, return 'Rutabagas! That doesn't work.'

Scope

Scope - where a variable is available in your code.

- Local Scope – available within a block or function
- Global Scope – available in main body of file.
- Execution Context – JS engine is wrapping our code in a context.
- https://www.udemy.com/course/understand-javascript/

JS engine


Execution Context is Created (CREATION PHASE)
Global Object
'this'
Outer Environment
"Hoisting"
Variables Setup
(and set equal to 'undefined')
and Functions Setup

Execution Scope

JS example-scope.js > ...

```
1   function b() {
2       let myVar;
3       console.log(myVar);
4   }
5
6   function a() {
7       let myVar = 2;
8       console.log(myVar);
9       b();
10  }
11  let myVar = 1;
12
13  console.log(myVar);
14  a();
15  console.log(myVar);
16
```

TERMINAL PROBLEMS OUTPUT DEBUG CONSOLE

```
[Running] node "/Users/caggarw/LaunchCode JS/scope/example-scope.js"
1
2
undefined
1

[Done] exited with code=0 in 0.102 seconds
```

Types

Primitive data types include:

1. string
2. number
3. boolean
4. undefined
5. null

Types example

Exceptions

Error Object – name, message

Types of Errors – SyntaxError, ReferenceError, TypeError

Throw

Default Error

Pre-existing Error

Custom Error – extending Error object.

Try catch

```
try {
//doSomething;
} catch(Error) {
//doSomething like log error
} finally {
//Optional cleanup or code.
}
```

Exercises

Write a function called divide that takes two parameters: a numerator and a denominator.

Your function should return the result of numerator / denominator.

However, if denominator is zero you should throw the error, "Attempted to divide by zero."

Exercises

Add a try/catch block inside of gradeLabs to catch an exception if the runLab property is not defined. If the exception is thrown, result should be set to the text "Error thrown".

Assignment 3 – Mar's rover (Curiosity).

due May 14

https://education.launchcode.org/intro-to-professional-web-dev/assignments/mars-rover.html

unit testing, modules, classes, array

Assignment 3 - Mars rover (Curiosity).

- Rover class – constructor and receiveMessage fn that takes in a Message.
- Command – has CommandType and value.
- Message object containing a name and Commands - Array of Command.
- Returns a Results – Array of objects with properties based on the commands.
- Properties in Rover object get updated based on the command.

Questions?

Studio time!