**Javascript Marathon Assignment – (Day – 2)**

**1. What is lexical structure?**

Lexical structure specifies set of some basic rules about how code should be written.

And it is a low-level syntax of a language.

* Rules like, how variables names should look like.
* The delimiter characters for comments.
* And the termination of a statement (; semi-colon).

**2. What is Unicode?**

* 1. Unicode formally, Unicode standard is an information technology standard for consistent encoding, representation and handling of text. The Unicode standard provides a unique number for every characters.
  2. Types – UTF-8, UTF-16 and UTF-32 (UTF-Unicode Transformation Format).

**3. Explain all the keywords present in the JavaScript with examples.**

1. switch, case, break, default
   1. **switch** – In switch, the case(s) is matched with expression then the statements inside the matched case(s) will executed otherwise if no case(s) matched with expression then the default’s statement will be executed.
   2. **case** – It is used to evaluate with the switch expression.
   3. **break** – It is used to terminate from the current loop.
   4. **default** – This clause will execute when no cases is matched.

**Example**:

const option = “javascript”;

switch (option)

{

case “c”:

console.log (“C Programming”);

break;

case “java”:

console.log (“Java Programming”);

break;

case “javascript”:

console.log (“Javascript Programming”);

break;

default:

console.log (“Other Programming language”);

}

1. try, catch, finally
   1. **try** – The try statement defines a code block to be run (to try).
   2. **catch** – The catch block specifies what to do if exception throw in try block

and used to observe errors. If exception is not throwed in try

then catch statement is skipped.

* 1. **finally** – The finally statements will be executed after try and catch block.

It always executed regardless whether exception is thrown or caught.

* 1. **throw** – It is used to throw custom errors.

**Example:**

try

{

display ();

}

catch(error)

{

// Throw Reference Error: display is not defined.

console. error(error);

}

finally

{

// This statement will be logged into console even error occurs in try block.

console.log ("This will work");

}

1. class

class in javascript is a blueprint for creating objects and it encapsulates data and function that manipulates data.

**Example:**

class Person

{

Constructor(name)

{

this.name = name;

}

getName ()

{

return this.name;

}

}

let obj = new Person(“Javascript”);  
console.log(obj.getName()); // Javascript

1. const

Constants are block scoped. The value of constant cannot be change.

(i.e) it cannot be reassigned or redeclared. If a constant is an object or array, their property or items can be updated or removed.

**Example:**

const num = 7;

num++;

console.log(num); // Type error : Assignment to constant variable

1. continue

The continue statement terminate the execution of the statement of the current iteration of the loop and continues the next iteration of the loop.

**Example:**

let i = 5;

for(i = 0;i<5;i++)

{

if(i == 3)

{

continue;

}

Console.log(i); // Output: 0124

}

1. debugger

This will invoke the available debugging functionality like setting a breakpoint, if there is no debugging functionality available then it has no effect.

**Example:**

function display()  
 {

/\* when debugger is invoked, then the execution is stopped in debugger statement. \*/

debugger;

console.log(“Hello Javascript”);  
 }

1. delete

The delete operator removes the property from an object.

**Example:**

const myobj =

{

firstname : "prithivi",

lastname : "raj"

}

delete myobj.firstname;

console.log(myobj); //{lastname:”raj”}

1. do and while
   1. **do** – do keyword is used along with while keyword to create a loop which executes a block of statements until the expression is evaluated to be false. It executes a block of statements at least once because it checks for condition after the execution of do block of statements.
   2. **while -** while create a loop which executes a block of statements as long as expression evaluate to be true. It evaluates an expression before each iteration.

**Example:**

let count = 0;

do

{

console.log(count); // output: 0 1 2

count++;

} while (count < 3);

1. if and else

The if block of statements executed if the condition is satisfied otherwise the statements inside the else block will executed.

**Example:**

const programming = “javascript”;

if (programming === “javascript”)

{

console.log (“Javascript Programming”);

}

else

{

console.log (“Other Programming Language”):

}

1. export and import

**export** – export declaration is used when creating modules, to export variables, objects, function from the module so that the other module can access them.

**Import** – To access the exported values by modules in another module we need to import them in module where we want to use.

**Example**:

export.js

const a =7;

export { a as “a-b”};

import.js

import { “a-b” as a } from export.js

1. extends, super and this

**extent** - The extends keyword is used with class declaration to create class that is child of another class.

**Super** - It is used to invoke the constructor of the super class.

**this –** this is referred to an object in javascript which depends on its invoked.

In the below example this referred to global object.

**Example:**

class parent // parent class

{

constructor(message)

{

this.message = message;

}

Display()

{

console.log(this.message);

}

class Child extends Parent // child class which extends parent class

{

constructor(mess,text)

{

Super(mess);

this.text = text;

}

result()

{

console.log(this.Display() + “ “ + this.text);

}

}

const obj = new Child(“hello”,”world”);

console.log(obj); //output : hello world

1. For

for creates a loop which consists of three expression enclosed in parentheses and separated by semicolons, followed by block of statements which will executed in loop.

**Example:**

for(let i = 0 ; i < 3; i++)

{

console.log(“Javascript”);

}

/\* output

Javascript

Javascript

Javascript

\*/

1. Function and return

**Function** - function is used to create a block of statement that is reused multiple times, function can be invoked by function name followed by parentheses and inside parentheses we can pass arguments to function parameters.

**Return** – The return statement ends the execution of function, specifies value to be returned and pass the control back to the function caller .

**Example:**

function add(a,b)

{

return a+b;

}

console.log(add(5,2)); //output: 7

1. In

The in operator return true if the specified property is present in specified object.

**Example:**

const car = { name:’Honda’, year:1998};

console.log (‘name’ in car); //output: true

console.log (‘type’ in car); // output: false

1. new

The new operator let allow as to create instance of user-defined and in-built object which has a constructor.

**Example:**

function programming(name,year)

{

this.name = name;

this.year = year;

}

const Programming = new Programming(“Javascript”,1995);

console.log (Programming.name); //output: Javascript

1. typeof

The typeof in javascript is used to find the type of a variable or values.

**Example:**

const num = 7;

console.log (typeof num); //output: number

console.log (typeof “javascript”); //output string

1. var

The var keyword is used to declare variable with function or global scope.

**Example:**

var num = 7;

if(num === 7)  
{

num=27;

console.log(num); // output:27

}

console.log(num); // output:27

1. void

The void evaluates the expression and return undefined.

**Example:**

const value = void 1;

console.log(value); //output: undefined

**4. What are shorthand operators, explain with a suitable example?**

The shorthand operators are used to perform arthimetic operation and storing the result to a variable in the same statement.

**Example:**

let a = 6;

a += 1; //which is equivalent to a = a + 1;

console.log(a); //output 7

**5. What is “use Strict” in JavaScript?**

* The purpose of using the use strict is to execute javascript code in strict mode.
* With strict mode we cannot able to access undeclared variables.

**Example:**

“use strict”

Pi = 3.14; // error Pi is not defined.