# JavaScript Class 1

Memorize primitive data types for job interviews.

SNOBNUS – string, number, object, Boolean, null, symbol.

Primitive Data Types: represent data in code

* String
* Numeric – Number data type for all numbers,
* Boolean
* Null
* Undefined

Modula: gives the remainder

25%5

Exponential : 2\*\*3 = 8

Order of operations: multiplication,

NaN—not a number 0/0 for example- saved as a value not error. If you add to a number will give NaN

Infinity

0

-0

# Variables

2 ways to create variables – new way to declare variables

Let

Const

## Let

Let age=72;

Use camelCase to name variables : numberOfHens

Let AvgRating = 9.7

Score+=1

Const hens=4 -🡪 cannot change the value, constant value.

Example: const pi=3.4159

Const daysOfTheWeek=7

## VAR 🡪 old way

Var tripDistance=100 ---- no need to use it, still works not ideal

# Boolean—true or false values , efficient

Let isLoggedIn=true;

Let isAngry=false

Can change the type of variable, not a good idea to do it

# String

Wrapped in quotes

Typeof () returns the type of variable

String index each character

String.length – returns length of string

Let mySong=”Surfin ‘ USA”

mySong[0]=S

cannot change strings by referencing characters

## string methods:

touppercase lowercase

trim() – removes spaces in string

color.trim().touppercase() - ---- can call more than one function

methods with arguments:

* “baseball”.indexOf(“ball”) -- return 4 where ball starts
* Index -1 means not there
* Slice takes slices of string – “Baseball”.slice(4) 🡪 “ball” does not change the variable. Slice can take two arguments (start, end)
* Replace(‘existing string’, ‘repalcement’)

Escape character : \” to show quotes

\n to print newline

\t—make a tab

\\ to show \

## String Template Literals

Allow embedded expressions to be evaluates and turned into a string

Use ` charachers under the esc

`I counted ${3+4} sheep`; // “I counted 7 sheep”

# Math Object()

# Functions

Function declaration

Function functionName(){

}

Function square(num)

{

}

Return ends function execution, code afterwards does not run!

Parameters: place values

Arguments: actual values

Declare a function inside a variable

Const myFunc=function (){

Return “hello”;

};

Const myaRRAY=[]

MYARRAY.PUSH(myobj)

Myarray.push(myfunc)

Console.log(myarray[1])

//can pass a new property to a function

Myobj.newProperty=”value” -🡪 key and value

A function can return another function

A function can be passed as parameter to another function.

Funnction expression : stored in variable

const square= function (num){

return num\*num;

}

Variable has no name, stored in variable

Const sum=function (x,y){

Return x+y;

}

Add(4,5) -- call

Console.dir(sum) – print object , proves it is an object

Const product= function multiply (x,y){ --🡪 can have a name, but use product to call it.

Return x\*y;

}

# Arrow Functions

Not supported in IE, work exactly the same way, but shorter—for one parameter the () are optional.

Const square= (x)=>{

Retur x\*x;

}

Const isEven=(num)=>{

Return num%2===0;

}

More compact ways to write functions:

Implicit rerurn—no need to write return keyword

Const square= n=>{

Return n\*n;

}

The above can be written as:

Const square = n=>(

N\*n;

)

Const square= n=> (n\*n); can even drop the ()

== 🡪 value not type 7==’7’ true

===🡪 value and type 7

Nul==undefined—true

Null===undefined—false

Go with === 99.9% of times

Same with !== better than !=

# Nov 9, 2021 JavaScript lecture 2

Data types, arrays,

Switch (variable)

Case (condition):

Procedure

Break;

Default – run at the end

Can include more than one case for the same condition

Case ‘sad face’:

Case ‘happy face’

Print “mesg”

Break;

## Ternary Operator – 3 pieces

Condition? expIfTrue: expIfFalse -- only simple yes/no

Num==7? Console.log(“lucky”): console.log(“unlucky”)