**Week-4 > Lecture 42, Part 1: Javascript Types**

1. **Object type**:-> It is defined by the name value pair.

**Ex:** name: ”chitransh”,

Social: {fb: “chiku”, linkedin: “chiku” }

1. **Boolean**:-> its value is true or false.
2. **Undefined**:-> It defines that no values has ever been set. You can define this variable as a value to undefined but you should not do that, as it is counter to undefined.
3. **NULL**:-> it signifies lack of value. Its ok to set a variable to null.
4. **Number**:-> it is the only numeric type in JS. It is always represented under the hood as double precision 64-bit floating point. JS does not have integer type.
5. **String** :-> It is sequence of characters as in other languages, represented in single and double quotes.

**Week-4 > Lecture 45, Part 1: Creating Object Using ‘new object()’ Syntax**

**Object in javascript->** object in JS is a collection of name-value pairs. As dictionary in python.

var company=new object(); // company object is created

company.name=”Facebook”;

Error->company.ceo.firstname=”Mark” // show error – bcoz here ceo is undefined and we are setting its firstname property.

Right way-> Setting of object within an object( here company is object and company.ceo is another object inside it)

company.ceo= new object();

company.ceo.firstname=”mark”;

For printing object in JS-> console.log(company);

There are 2 ways to access object property in JS->

1. By using .(dot operator)-> company .name=”facebook”
2. By using bracket[] -> console.log(company[name])

var stockpropname=”stock of company”

company[stockpropname]=220

console.log(“stock price is:” + company[stock of company])

**Week-4 > Lecture 45, Part 2: Creating Object Using literal syntax**

var facebook = {

name : "Facebook",

ceo :{

firstname : "mark",

favcolor : "blue"

},

"stock of company" : 110

};

console.log(facebook.ceo.firstname);

console.log(facebook);

**Note**- Accessing the elements is same in.

**Week-4 > Lecture 46, Functions Explained**

Functions in JS is a object that have special properties to them.

Function syntax-

Function multiply(x,y){

return x\*y;

}

Console.log(multiply(5,3))

#setting property on multiply object as we do in other objects-

multiply.version= “v1.0.5.4” / this states that fuction is a type of object.

console.log(multiply.version)

# Factory Function-> When a function return an object, we call it factory function. It can produce object instance without ‘new’ keyword or ‘classes’.

Syntax->

function makemultiplier(multiplier){

var myfunc= function(x){

return multiplier \*x;

};

return myfunc;

}

Calling of factory function-

var multiplyby3= makemultiplier(3)

console.log(multiplyby3(10)) -------🡪output=30

# Passing function as arguments-

Syntax->

Function doOperatioOn(x,operation){

return operation(x);

}

var result= doOperationOn(5,multiplyby3); // here we are passing the reference of multiplyby3

console.log(result); // -------🡪 output: 15

**Week-4 > Lecture 47, Part1: Pass by value,pass by reference**

In JS, premittives are pass by values, and objects are passed by reference.

#pass by value-

X=5;

Y=x;

Console.log(y) // both value are same. If we change anyone, other will not be affected.

#pass by reference-

Value={x:5};

Y=value // here both(value and Y)are pointing to same memory location. Therefore if anyone of them is changed then other will also change.

**Week-4 > Lecture 48, Functions, Constructors, Prototypes and ‘this’ Keyword**

Practical in Brackets.

**Week-4 > Lecture 50, Part1: Arrays**

Two methods of creating arrays:

1. Using new keyword:->

var array = new Array();

array[0]= “chiku”;

array[1]= 2;

array[2]= function(name){

console.log(“Hello”+ name );

};

array[3]= {course: ”html,css,js” }

# Access the arrays:

Console.log(array);

array[2](array(0)); // function will call.

console.log(array[3].course);

Note: we have multiple types of elements in an array.

1. Short hand array creation:

var names = [“chiku”, “krishna”, “ronak”];

console.log(names);

# To iterate over an array-

for (var i=0; i< names.length; i++){console.log(names[i]);

}