Node JS

Part 1 : DISCOVER JS

DISCOVER JS

1. Declaring variables
2. Primitive Types
3. Declaring constants
4. Simple manipulations and operations
5. Conditions & Loops
6. Arrays
7. Functions
8. Objects
9. More Arrays
10. Lambda

1. Declaring variables

```
1  // -Declarations
2  // var X
3
4  var name;
5  var anotherName;
6  var Name3;
7  var 4thName; // wrong declaration
8  var diffrent_name;
9  var $erfez; // wrong declaration
10 var $name;
11 var _name;
```

2. Primitive Types

```
1  // Types
2  var
3  // Numbers
4          num = 5,
5          float = 6.3,
6  // strings
7          text = "message",
8          text2 = 'message2',
9  // bool
10         conditionT = true,
11         conditionF = false,
12 // undefined and null
13         empty1= null,
14         empty2= undefined;
```

3. Declaring Constants

```
1 // Constants
2 const Trainer= "Youcef Madadi";
3 const Students= 30;
4
5 Trainer = "Abdelhak Ihadjadan" /* Uncaught TypeError: Assignment
6                                 * to constant variable.*/
7
8 const Assistant; /* Uncaught SyntaxError: Missing initializer
9                   * in const declaration*/
```

4. Simple manipulations and operations

Arithmetic operations

```
1 // Arithmetic operations
2 x = 5 + 6; // addition
3 y = 7 - 3; // minus
4 z = 4 * 4; // multiplication
5 a = 7 / 2; // division
6 b = 4 ** 2; // power
7 b = b % 2; // Modulus
8 a++; // Increment
9 b--; // Decrement
```

4. Simple manipulations and operations

String manipulation

```
1 // String manipulation
2 name = "Youcef";
3 text = "Hello new user : " + name;
4 message= `From ${name} : ${text}`
```

4. Simple manipulations and operations

Logical operations

```
1  // Logical operations
2  c = (x >= y); // ( x > y )
3  d = (y <= z); // ( y < z )
4  e = (z == x);
5  f = (5 == "5");
6  g = (5 === "5");
7  h = (5 !== "5");
8  i = (name == text);
9  j = (!g);
10 k = (c && e);
11 l = (c || e);
```

4. Simple manipulations and operations

Binary operations

```
1 // Binary operations
2 m = (7 & 6);
3 n = (10 | 9);
4 o = (~z);
5 p = (x ^ z);
6 q = (5 >> 1);
```

5. Conditions & Loops

Conditions

```
// Conditions
if (/* condition1 */) {
  /* Instractions 1 */
} else if (/* condition2 */) { // optional
  /* Instractions 2 */
} else { // optional
  /* Instractions 3 */
}
```

Conditional (ternary) operator

```
// question mark operation (ternary)
// ( condition ) ? (true Result) : (false Result)
var max = a>b ? a : b;
```

5. Conditions & Loops

Switch case

```
// Switch
switch(Bounadem){
  case Trainer:
    console.log("You can Teach go on");
    break;
  case Assistant:
    console.log("Well he gotta take care of the trainer");
    break;
  case CoAssistant:
    console.log("You can just sell della3")
    break;
    default:
      console.log("You are a trainee");
      break;
}
```

5. Conditions & Loops

Loops (while)

```
1 // loop 1
2 while (/* condition */) {
3   /* Instractions to repeat */
4 }
```

Loops (do while)

```
1 // loop 2
2 do{
3   /* Instractions to repeat */
4 } while  (/* condition */) ;
```

5. Conditions & Loops

Loops (for)

```
// loop 3
for (var i = 0; i < N; i++) {
  /* Instractions to repeat */
}
```

Loops (for reverse)

```
// loop 4
for (var i = N; i > 0; i--) {
  /* Instractions to repeat */
}
```

6. Arrays

Declaration

```
1 // Array Declarations
2 var list=[];
3 var Notes=[12,14,8,16,4,9,13,12,"Absent"];
```

Assignment

```
1 // Array Assignment
2 list[0]="Youcef";
3 list[4]="Abdelhak";
```

7. Functions

Declaring a function

```
1 // Declaring a function
2 function Max(a, b) {
3   if (a > b) return a;
4   else return b;
5 }
```

Calling a function (invoking)

```
1 // Declaring a function in a variable
2 var a=prompt("enter a"),
3     b=prompt("enter b");
4 console.log( Max( a , b ) );
```

7. Functions

Declaring a function in a variable

```
// Declaring a function in a variable
var maxf = function(a,b){
            if(a>b) return a;
            else return b;
          }
```

Name of a function

```
// name of a function
console.log( Max.name , maxf.name );
```

7. Functions

Sending a function as a parameter

```
// Sending function as a parameter
function call(Caller,a,b){
  return Caller(a,b);
}
```

Arguments for the function

```
// not setting any arguments and getting away with it
function Max(){
        var max=-Infinity;
        for(var i=0; i < arguments.length ; i++){
                max=Math.max(max,arguments[i]);
        }
        return max;
}
```

8. Objects

Declaring an object

```
1 // Declaring an object
2 var person={
3   first_name:"Youcef",
4   "last name":"Madadi",
5   age:23
6 }
```

Accessing Data in an Object

```
1 // Accessing data in objects
2 console.log(
3         person.first_name,
4         person["last name"],
5         person["age"]
6 )
```

8. Objects

Get Objects properties

```
// Getting Keys in an object
for(var key in person){
  console.log(key);
}
```

Getting Keys

```
// Getting Keys of an object
Object.getOwnPropertyNames(person)
Object.keys(person)
```

8. Objects

Methods

```

// puting functions in objects (methods)
var person1 = {
  first_name: "Youcef",
  "last name": "Madadi",
  age: 23,
  ToDo: function (Do) {
    return `I'm Going to do : ${Do}`;
  },
};
```

8. Objects

this key word

```
// this key word to access other members
var person2 = {
  first_name: "Youcef",
  "last name": "Madadi",
  age: 23,
  intreduce: function () {
    console.log(
      `I'm ${this.first_name} ${this["last name"]} and i'm ${this.age} years old`
    );
  },
};
```

8. Objects

Constructing Object (new key word)

```
1  // Genrate Objects (classes first form)
2  function Person(first_name, last_name, age) {
3    this.first_name = first_name;
4    this.last_name = last_name;
5    this.age = age;
6    this.fullName = function () {
7      return this.first_name + " " + this.last_name;
8    };
9  }
10 var p1 = new Person("Youcef", "Madadi", 23);
11 var p2 = new Person("Abdelhak", "Ihadjadan", 22);
```

9. More Arrays

Length

```
// Getting Keys in an object
Peoples=[p1,p2,p3...]
console.log(Peoples.length)
```

Checking arrays

```
// checking a variable if it's an array
if(Array.isArray(Peoples)) return Peoples;
else return [];
```

10. Lambda functions

```
// Lambda functions
var sum = function (a, b) {
  return a + b;
};
// we could write the above example as:
sum = (a, b) => a + b;
// or
sum = (a, b) => {
  return a + b;
};

var max = (a, b) => (a > b ? a : b),
  fact = (n) => (n > 1 ? n * fact(n - 1) : 1);
```