# AJS Question Answer

## 1. How can you create an object in javascript? explain with an example.

Ans. In JavaScript, an object can be created in two ways:

(1) Using Object Literal / Initializer Syntax.
(2) using the Object() Constructor function with the new keyword.

Example using Object literal:

```
const person = {
        firstName: “Hunain”,
        lastName: “Chhipa”,
}
```

In this example, we create an object called person using an object literal, which is a comma-separated list of key-value pairs enclosed in curly braces {}. The keys are strings that represent property names, and the values can be any valid JavaScript data type, such as strings, numbers, booleans, arrays, or even other objects.

Example using Object() Constructor:

In JavaScript, you can create objects using the Object() constructor function with the new keyword. The Object() constructor is a built-in constructor function provided by JavaScript.

```
const person = new Object();
person.firstName = “Hunain”;
person.lastName = “Chhipa”;
```

In this example, we create an object called person using the Object() constructor with the new keyword. The new keyword creates a new instance of the Object constructor, and the resulting object is stored in the person variable.

## 2. How can you create an Array in javascript? explain with an example.

Ans. Arrays are used to store multiple values in a single variable. This is compared to a variable that can store only one value. Each item in an array has a number attached to it, called a numeric index, that allows you to access it. In JavaScript, arrays start at index zero and can be manipulated with various methods.

Example:

```
const data =  [“Brian”, “Dom”, 10, true, ];
```

### 3. What is Callback ? Explain with an example.

Ans. Any function that is passed as an argument to another function so that it can be executed in that other function is called a callback function.

Example:

```
function Demo(data) {
        console.log(data);
    }
    function Addition(num1, num2) {
        return num1 + num2;
    }
    const mydata = Addition(3, 2);
    Demo(mydata);
```

### 4. What are the ways to define a variable in Javascript ?

Ans. In JavaScript, you can declare a variable in different ways by using different keywords. Each keyword holds some specific reason or feature in JavaScript. Basically, we can declare variables in three different ways by using **var**, **let** and **const** keywords. Each keyword is used in some specific conditions.

**var:** This keyword is used to declare variables globally. If you used this keyword to declare a variable then the variable can be accessible globally and changeable also. It allows us to re-declare and re-assign the value.

*var x = 5; // declares a variable named x and assigns it the value 5*

**let:** This keyword is used to declare variables locally. If you used this keyword to declare a variable then the variable can be accessible locally and it is changeable as well. It allows us to re-assign the value.

*let y = 10; // declares a variable named y and assigns it the value 10*

**const:** This keyword is used to declare variables locally. If you use this keyword to declare a variable then the variable will only be accessible within that block similar to the variable defined by using let and the difference between let and const is that the variables declared using const values can’t be reassigned. So we should assign the value while declaring the variable.

*const z = 15; // declares a constant variable named z and assigns it the value 15*

## 5. What is the difference between null and undefined?

Ans. **Undefined:** Undefined means the variable has been declared, but its value has not been assigned. The typeof() operator returns undefined for an undefined variable.

*let myVar; // myVar is declared but not assigned a value, so it is undefined*

**Null:** Null means an empty value or a blank value. The typeof() operator returns the type as an object for a variable whose value is assigned as null.

*let myVar = null; // myVar is explicitly set to null*

## 6. What is NaN in javascript? Any base to integer in javascript?

Ans. In JavaScript, NaN stands for "Not-a-Number".In JavaScript, NaN is a number that is not a legal number.The Number.isNaN() method returns true if the value is NaN, and the type is a Number.

Here are some examples when `NaN` might be generated.

```
let result1 = 0 / 0; // Division by zero

console.log(result1); // Output: NaN


let result2 = "hello" - 5; // Subtraction of a non-numeric string from a number

console.log(result2); // Output: NaN


let result3 = Math.sqrt(-1); // Square root of a negative number

console.log(result3); // Output: NaN
```

In JavaScript parseInt() function is used to convert the passed-in string parameter or value to an integer value itself. This function returns an integer of the base which is specified in the second argument of the parseInt() function.

## 7. When should I use the Arrow function in Javascript?

Ans. Arrow functions introduce concise body syntax, or implicit return. This allows the omission of the curly brackets and the return keyword. Implicit return is useful for creating succinct one-line operations in map , filter , and other common array methods.

## 8. Explain the Prototype with an example.

Ans. JavaScript is a prototype based language, so, whenever we create a function using JavaScript, JavaScript engine adds a prototype property inside a function, Prototype property is basically an object (also known as Prototype object), where we can attach methods and properties in a prototype object, which enables all the other objects to inherit these methods and properties.

## 9. What is the difference between .forEach loop and .map loop?

Ans.

- Return Value: The main difference between forEach and map is that forEach does not return any value, whereas map returns a new array with the results of applying a provided function to each element of the original array.
- Mutability of Original Array: forEach operates directly on the original array and does not create a new array. It is used when you want to perform an action for each element in the array, such as modifying the elements in place, but it does not return a new array with the results.
- Immutability of Original Array: On the other hand, map creates a new array with the results of applying a provided function to each element in the original array. The original array is not modified, and a new array with the same length as the original array is returned.

## 10. What is a constructor in javascript? Explain with example.

Ans. In JavaScript, a constructor is a special type of function that is used to create and initialize objects. Constructors are called with the new keyword, and they are used as blueprints or templates for creating multiple instances of objects with similar properties and methods.

Example:

```
function Demo(fname, lname) {

      this.fname = fname;

      this.lname = lname;

  }

  const data = new Demo("Hunain", "Chhipa");   console.log(data);
```

### 11. What is the role of closure in javascript?

Ans. **Closure**: Closures are created when a function is defined inside another function, and the inner function refers to variables from its outer function, even after the outer function has returned. Here's an example:

### 13. What is the output of below code?

```
var array = ["DataFlair", 2019, 1.0, true];

var msg = "Array: [";

for(var i = 0; i < array.length-1; i++){

msg += array[i] + ", ";

}

msg += array[array.length-1] + "]";

console.log(msg);
```

Ans. Array: [DataFlair, 2019, 1, true]

### 14. What is meant by “this” in javascript?

Ans. In JavaScript, this is a special keyword that refers to the current object or the object that is currently being executed or accessed within a function or method. The value of this depends on how and where a function or method is called, and it is determined dynamically at runtime.

### 15. How to validate a form in javascript?

Ans. Validating a form in JavaScript typically involves checking that the data entered by a user conforms to the expected format or meets certain criteria before it is submitted to a server for processing.

## 16. What are object prototypes?

Ans.In JavaScript, object prototypes are a mechanism that allows objects to inherit properties and methods from other objects. A prototype is an object that serves as a template or blueprint for creating other objects. Every object in JavaScript has a prototype, and objects created from the same constructor function share the same prototype.

## 17. What is the rest parameter?

Ans. The rest parameter is prefixed with three dots (...). The rest parameter is an improved way to handle function parameters, allowing us to more easily handle various inputs as parameters in a function. The rest parameter syntax allows us to represent an indefinite number of arguments as an array. With the help of a rest parameter, a function can be called with any number of arguments, no matter how it was defined.

## 18. What is the use of promises in javascript?

Ans. A Promise in JavaScript is an object that holds the future value of an asynchronous operation. For example, if we are requesting some data from a server, the promise promises us to get that data that we can use in the future.

A promise object can have the following states:

- Pending: it is an initial state, where the result is not ready, and it is waiting for the asynchronous operation to get finished.
- Resolved: it means that performed action completed successfully. i.e., Function returned the promised value.
- Rejected: it means that performed action failed or an error occurred. i.e., Function did not return the promised value.

Syntax :

```
var promise = new Promise(function(resolve, reject){

//do something

});
```

## 19. What are classes in javascript?

Ans. Class is a template for javascript objects.They encapsulate data with code to work on that data. Classes in javascript are built on prototypes but also have some syntax and semantics that are unique to classes.

Example:

```
class myName {
    constructor(fname, lname) {
        this.fname = ‘Hunain’;
        this.lname = ‘Chhipa’;
    }
}
const fullName = new myName;
console.log(fullName);
```