1. **What are ways to create the objects?**
   1. Using object literal
   2. With new keyword
   3. Using a constructor

Eg:

a:

var person = {firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"};

b:

var person = new Object();

person.firstName = "John";

person.lastName = "Doe";

person.age = 50;

person.eyeColor = "blue";

c:

function person(first, last, age, eye) {

this.firstName = first;

this.lastName = last;

this.age = age;

this.eyeColor = eye;

}

var myFather = new person("John", "Doe", 50, "blue");

var myMother = new person("Sally", "Rally", 48, "green");

1. **How many ways we can create the arrays?**
   1. Array literal
   2. Using new keyword

Eg:

a:

var cars = [  
    "Saab",  
    "Volvo",  
    "BMW"  
];

b:

var cars = new Array("Saab", "Volvo", "BMW");

1. **What are the arguments in javascript functions?** 
   1. Arguments are the parameters in js functions, which are real values received by the function when it is invoked. Inside the function they behave as local variables

Eg:

function name(parameter1, parameter2, parameter3) {  
    code to be executed  
}

1. **What is prototypal inheritance in javascript?**
   1. All js objects inherit their properties and methods from their prototype. For instance, objects created with new Date() inherit the Date.prototype
   2. The Object.prototype is on the top of the prototype chain

Eg:

function Person(first, last, age, eyecolor) {  
    this.firstName = first;  
    this.lastName = last;  
    this.age = age;  
    this.eyeColor = eyecolor;  
}

now adding a new property in person object using a prototype

Person.prototype.nationality = "English";

1. **What are enumerators in javascript?**
   1. Enumerators are json like object in js except the properties’ values are not in quote. Enum properties can be accessed using dot (.) notation like in json.

Eg:

**var** SizeEnum = {  
  SMALL: 1,  
  MEDIUM: 2,  
  LARGE: 3,  
};

Then use it like so:

**var** mySize = SizeEnum.SMALL;

1. **Callbacks and Closures?**
   1. Callbacks are the function that contains parameters as another function
   2. A closure is a function having access to the parent scope, even after the parent function has closed.

Eg:

a:

function bindEvents() {

var countryDdl = controls().countries;

countryDdl.addEventListener("change", function () {

buildStates(this.value).us();

});

}

b:

var add = (function () {  
    var counter = 0;  
    return function () {return counter += 1;}  
})();

1. **Module based programming in javascript.**

Js support the oop concept so we can call it module based programming. Each component can be defined separately and can be used in multiple places in the code. So, the reusability of the component implies that it is modular.

1. **What is strict mode in javascript?**

* The purpose of "use strict" is to indicate that the code should be executed in "strict mode".
* With strict mode, you can not, for example, use undeclared variables.

Eg:

“use strict”

X = 3.14; // this will cause an error b/c x is not defined

1. **What is the difference between == and ===**
   1. == equality check, === checks equality and value type also

var a = 5;

var b = "5";

var c = (a==b); // true

var d = (a===b); // false

1. **Ternary operator**
   1. The ternary (?) operator can be used as a shortcut for an if...else statement

**Eg:**

var now = new Date();

var greeting = "Good" + ((now.getHours() > 17) ? " evening." : " day.");

1. **Difference between public, private and static variables and their use cases**

Public: Any function can access, modify, or delete those members, or add new members.

Eg: function Container(param) {

this.member = param;

}

var container = new Container(“a”);

// now member can be accessed like this:

Container.member // returns a

Private: only private function can access them

Eg:

function Container(param) {

var a = 5;

// now a is not accessible using //Container.a, however, private function can access it.

Function add(){

// now add can access a

Var sum = a + 5;

}

}

Static: static variables are shared by all instances of the object, useful scenario would be counting example

Eg:

function Person(){

this.name = "Peter";

// counter is a static variable

Person.counter = Person.counter + 1;

}

Person.counter = 0;

var p1 = new Person;

var p2 = new Person;

console.log(Person.counter); // 2