**Ques. What is Hoisting ?**

Ans - Hoisting is the JavaScript interpreter’s action of moving all variable and function declarations to the top of the current scope. However, only the actual declarations are hoisted. Any assignments are left where they are.

e.g - <https://www.sitepoint.com/back-to-basics-javascript-hoisting/>

**Ques. What is difference between == and ===**

Ans – Both is comparision operator but == compares the value but === compares value and type both.

e.g – “5”==5 : true

but “5”===5 : not true

**Q. What is closure ?**

**Ans -** A closure is a function, along with all variables or functions that were in-scope at the time that the closure was created. In JavaScript, a closure is implemented as an “inner function”; i.e., a function defined within the body of another function. Here is a simplistic example:

e.g - function showName(firstName, lastName) {

var nameIntro = "Your name is ";

function makeFullName() {

return nameIntro + firstName + " " + lastName;

}

return makeFullName();

}

nameIntro is closure for makeFullName.

showName("Michael", "Jackson"); // Your name is Michael Jackson

**Ques . What is iife in js ?**

**Ans -** It’s an [Immediately-Invoked Function Expression](http://benalman.com/news/2010/11/immediately-invoked-function-expression/), or [**IIFE**](https://en.wikipedia.org/wiki/Immediately-invoked_function_expression) for short. It executes immediately after it’s created.

It has nothing to do with any event-handler for any events (such as document.onload).  
The first pair of parentheses **(**function(){...}**)** turns the code within (in this case, a function) into an expression, and the second pair of parentheses (function(){...})**()** calls the function that results from that evaluated expression.

This pattern is often used when trying to avoid polluting the global namespace, because all the variables used inside the IIFE (like in any other normal function) are not visible outside its scope.  
This is why, maybe, you confused this construction with an event-handler for window.onload, because it’s often used as this:

(function(){

// all your code here

var foo = function() {};

window.onload = foo;

// ...

})();

// foo is unreachable here (it’s undefined)

**Correction suggested by [Guffa](https://stackoverflow.com/users/69083/guffa):**

The function is executed right after it's created, not after it is parsed. The entire script block is parsed before any code in it is executed. Also, parsing code doesn't automatically mean that it's executed, if for example the IIFE is inside a function then it won't be executed until the function is called.

**Q. Difference Between var and let keyword ?**

**Ans –**

1. Var keyword was since the beginning of js but let is introduceted in ecmascript6.
2. The variable declared with var are hoisted but with let are not and that is the reason you can access the variable declare with var before its initialization.

**Q. Difference b/w let ans const?**

**Ans –** variable declared with let can be reassigned but with const can’t.

**Q. Difference between undefined and null ?**

**Ans –** both represents the empty value. But when variable is declared and not assigned, they have default value defined whereas null is assigned manually to a variable to represent empty value.

**Q. What is Prototype Inheritance ?**

**Ans -** All JavaScript objects inherit properties and methods from a prototype.

Date objects inherit from Date.prototype. Array objects inherit from Array.prototype. Person objects inherit from Person.prototype.

The Object.prototype is on the top of the prototype inheritance chain:

Date objects, Array objects, and Person objects inherit from Object.prototype.

Sometimes you want to add new properties (or methods) to all existing objects of a given type.

Sometimes you want to add new properties (or methods) to an object constructor.

The JavaScript prototype property allows you to add new properties to object constructors:

Example

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

**Q. D/f between function declaration and function definition?**

**Ans –**

function fd(){

console.log(‘declaration’);

}

Var fE = function(){

Console.log(‘function expression’);

}

Function declaration can be used before its declaration but not the expression since it’s a variable so its behaves like a variable.

**Q. What is promise and why do we use it ?**

**Ans –** Promise is a way to execute asynchronous functions. As the name says – it behaves like a promise .  
e.g – my mom says if I will clean my room, he will let me play the game. Now my mom has done a promise and which will be resolved if I will clean the room otherwise she would just throw a shoe at me which is error .

#### Q. Can you name two programming paradigms important for JavaScript app developers?

Ans A multi-paradigm language, JavaScript supports object-oriented, imperative, and functional programming styles. Javascript support oops with prototypal inheritance.

***Q: What is the significance of including 'use strict' at the beginning of a JavaScript source file?***

Though there is much more to be said on the topic, the short and most important answer here is that use strict is a way to voluntarily enforce stricter parsing and error handling on your JavaScript code at runtime. Code errors that would otherwise have been ignored or would have failed silently will now generate errors or throw exceptions. In general, it is a good practice.