JS Interview Question

Q) What is Hoisting?

By) It refers to the process whereby the interpreter appears to move the declration of functions, variables or classes to the top of their scope, prior to the execution of the code. Hoisting allows functions to be safely used in code before they are declared.

Q) What is Scoping?

By) Scoping is determining where variables, functions and objects are accessible in your code during run-time. This means the scope of a variable (where it can be accessed) is controlled by the location of the variable declration.

In JS, there are two scope:

1) Global Scope
2) Local Scope

Q) How are var, let and const are diffrent?

By) The scope of var is function scope whereas let & const are block-scope. var can be updated & re-declared into the scope. Where, let can be updated but can't be redeclared into the scope and const neither be updated nor re-declared into the scope. Var & let both can be declared without initialization where const can't be declared without initialization.

Q) What are two main differences in arrow functions?

Ans) Unlike regular functions, arrow functions do not have their own this. The value of this inside an arrow function remain the same throughout the lifecycle of the function and is always bound to the value of this in the closest non-arrow parent function.

Arrow functions do not have an arguments binding. However, they have access to the arguments object of the closest non-arrow parent function. Named rest parameters are heavily relied upon to capture the arguments passed to arrow functions.

Regular function created using function declration or expressions are constructable and callable. Since regular functions are constructable, they can be called using new keyword. However, the arrow functions are only callable not constructable.

Q) Does call, apply, bind work for arrow functions?

By) In case of arrow functions our methods. call, apply and bind doesn't work. Since arrow function do not have their own this, the methods call() or apply() can only be pass in parameters. ~~this~~ thisArg is ignored.

Q) What does call, apply, bind do?

Ans) Call, apply, bind are all methods ~~each~~ within the function prototype. What they actually do is allows us to call a function with a given this context and arguments. They let us call a function and have access to the properties of another function or object. We can borrow methods of one object's prototype and use it to another. For example we could apply Array's Slice method to a string or use Math's max to find the maximum number of a bunch of numbers in an Array.

Call → It accepts this and a list of arguments

e.g. let obj = {food: "Pizza"}

function saveFood(text){
return `${this.food} ${text}`;
}
let text = "is awesome!"
saveFood.call(obj, text); //O/P → "Pizza is awesome!"

Apply → Apply is same as call but instead, apply accepts a single array of arguments.

Bind → The tricky part about bind is that it has the same functionality as apply but instead of calling the function immediately, it returns a bound function.

Q) What are closures?

Ans) A closure is the combination of a function bundled together (enclosed) with references to its surrounding state (the lexical environment). In other words, a closure gives you access to an outer function's scope from an inner function. In JS, closures are created everytime a function is created, at function creation time

Q) What is Event & Bubbling?

Ans) Event Bubbling is a method of event propagation in the HTML DOM API when an event is in an element inside another element, and both elements have registered a handle to that event. It is a process that starts with the element that triggered the event and then bubbles up to the containing elements in the hierarchy. In event bubbling, the event is first captured and handled by the innermost element and then propagated at outer elements.

Syntax: addEventListener(type, listener, useCapture)

type: use to refer to the type of event.

listener: Function we want to call when the event of the specified type occurs.

useCapture: Boolean value. Boolean value indicates event phase. By default, it is false. It means it is in the bubbling phase.

Q) What is Event loop?

Ans) JS has runtime model based on an event loop, which is responsible for executing the code, collecting and processing events, and executing queued sub-tasks. This model is quite different from models in other languages like C and Java.

Q) Explain Promise to a 5 year old, with simple Examples.

Ans)

Q) What does async await mean?

Ans) Async/Await is used to work with Promises in asynchronus functions. It is basically syntatic sugar for Promises. It is just a wrapper to ~~restat~~ restyle the code and make Promises easier to read and use. It makes asynchronus code look more like Synchronus/Procedural code, which is easier to understand.

Q) What is Currying?

Ans) currying is the transformation of a function with multiple arguments into a sequence of single-argument function.