- **Closures:**

A closure in JavaScript is a function that has access to variables from outer scope to inner scope of a function.

Example for closure:

```
function demo()
{
   let n=5
   function demo1()
   {
      console.log(n)
   }
   return n
}
```

In the above code, variable n value can be accessed by the child function i.e., In **demo1()** Variable n value is accessed from **demo()**

- **Scope Chain :**

Whenever we try to access a variable it looks in current local scope if it is not present it goes to function scope like wise it goes to global scope until variable is found. This hierarchical structure determines the order in which Javascript looks for variables is called a Scope Chain.
Example :

```
function outer() {
  var innerVariable = 1;

  function inner() {
    console.log(innerVariable); // 1
  }

  inner();
}

outer();

// innerVariable is not accessible here because it is in the local scope of the outer() function
console.log(innerVariable); // ReferenceError: innerVariable is not defined
```

- **Lexical Environment :** Stores the variables and functions that are defined in the current scope and all of the outer scopes. It is also known as the lexical scope or the lexical closure.

  The lexical environment is created when a function is called and destroyed when the function returns.