Promise:

Explain flow of the below code

```
function job() {
   return new Promise(function(resolve, reject) {
      reject();
   });
}
let promise = job();
promise
.then(function() {
   console.log('Success 1');
})
.then(function() {
   console.log('Success 2');
})
.then(function() {
   console.log('Success 3');
})
.catch(function() {
   console.log('Error 1');
})
```

```
.then(function() {

   console.log('Success 4');

});
```

Explanation:

The process starts by defining a function named job(), which creates a new Promise. Inside this Promise, two functions are defined: one for resolving and one for rejecting. In this case, the promise is immediately rejected.

Subsequently, a variable named promise is initialized, and the result of calling job() is assigned to it. This variable now holds a Promise object.

Multiple .then() and .catch() methods are chained to the promise object to handle its resolution and rejection.

The first .then() method executes if the promise resolves successfully (i.e., if resolve() is called), logging 'Success 1' to the console.

The second .then() method follows the first one, logging 'Success 2' to the console.

The third .then() method is chained to the second one, logging 'Success 3' to the console.

If the promise is rejected (i.e., if reject() is called), the .catch() method is invoked, logging 'Error 1' to the console.

Finally, another .then() method is called, irrespective of whether the promise resolves or rejects, logging 'Success 4' to the console.

In summary:

- If the promise is immediately rejected within the job() function, it bypasses all .then() methods and proceeds directly to the .catch() method, logging 'Error 1' to the console.
- Following the .catch() method, it moves to the last .then() method, logging 'Success 4' to the console.