1. closures
2. Debouncing

Execution Context

creation

Execution.

global.

var a = 5

{

let a = 70

outer = function

```
function outer() {
  let a = 5;

  function inner() {
    console.log(a);
  }
  inner();
}

outer();
```

console.log(a)

3

function context

Vara = undefined

inner = function.

a = 5

✓ FC

5

```
55
56    //   output();
57
58    function authentication() {
59      let userName = "masai";
60      let password = "123";
61
62      return (user_password, user_username) => {
63        if (user_username === userName && user_password === password) {
64          console.log("access granted");
65        } else {
66          console.log("data is not matching");
67        }
68      };
69    }
70
71    let checking_data = authentication();
72
73    checking_data("masai", "123");
74
75    checking_data("tushar", "abc");
76
77    //   A inner function which can have access to outer functions variables and parameter even after the outer function is
      return is called closure.
78  </script>
79
```

It is used to reduce the number of feture request.

```
let a = 10

function outer(){

    let a = 5

    function inner(){

        console.log(a)

    }

}
```

?