```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>super</title>
</head>

<body>

    <script>
        /*
,            ***** extend ****
,
,            이제 'admin'은 'user'의 extension이다.
,            여기서 문제점은, 나는 더 많은것들을 추가하게 하고 만들고 싶다.
,            'admin'은 'user'이지만 user 이상이기 때문.
,
,            예를들어 'super admin의 true/false를 생성해보자'
,            constructor(superAdmin) {
,                this.superadmin = superAdmin
,            }
,            admin안에 이런식으로 constructor를 정의해주면, 기존의 constructor를 잃어버린다.
,            그래서 'admin'은 name부터 password까지가 뭔지 모르고있다.
,
,            먼저 user class를 리팩토링 먼저 할것이다.
,            왜냐하면 여기서(인자 받는부분) 유저클래스가 값들을 많이 가질것이기 때문.
,            options 오브젝트로 넣는것이 좋을듯하다.
,
,            const sexyUser = new User({
,                username: "bibi",
,                lasttName: "bucky",
,                email: "barnesquiat@gmail.com",
,                password : 1111
,            });
,            만약 내가 여러 arguments를 가지고 있다면 opttions오브젝트로 하는것이낫다
,            어떤 옵션을 넘겨주는지 볼 수 있기 때문.
,
,            constructor({ username, lastName, email, password}) {
,                this.username =  name;
,                this.lastName = lastName;
,                this.email = email;
,                this.password = password;
,            }
,            이런식으로 보내주어도 결과는 같다.
,
,            이전에 우리가 가졌던 문제점은 바로 admin을 만들고 싶었던것이다.
,            user의 data와 몇가지 새로운 데이터를 포함해서.
,            하지만 admin 안에서 새로운 constructor를 정의하니
,            기존에 갖고있던것이 없어져서 오류가났다.
,
,            만약 우리가 constructor없이 admin을 만든다면 상관이없다.
,            왜냐하면 기존에 가지고있는것을 무시하지 않기때문.
,            하지만 새로 만든다면 user constructor를 잃는다.
,
,            ***** super *****
,            이럴때에 특별한 super함수를 호출하는것이다.
,            이 함수는 classes안에서만 유효하다.
,            super가 무엇을하느냐.
```

```
,            super는 호출할것이다. constructor의 Base class(원시 클래스, 현재는 user)
,
,            class Admin extends User{
,                constructor({ username, lastName, email, password, superAdmin,
isActive}) {
,                    super({ username, lastName, email, password });
,                    this.superAdmin = superAdmin;
,                    this.isActive = isActive;
,                }
,                deleteWebsite() {
,                    console.log('Deleting the whole website')
,                }
,            }
,            const admin = new Admin({
,                username: "bibi",
,                lastName: "bucky",
,                email: "barnesquiat@gmail.com",
,                password : 1111,
,                superAdmin : true,
,                isActive : false,
,            })
,            console.log(admin)
,
,            우리는 이 admin에게 user의 옵션을 주어야하는데
,            기존의 넘겨받던 arguments에 추가로 내가 몇가지 options들을 정의하고
,            super에 base class의 constructor정보를 넘겨줄것이다.
,            그러면 super 는 base class인 user의 constructor를 호출할것이다.
,            그럼 기존에 갖고있던 username,lastName .. 을 넘겨받는다.
,
,            class는 자바스크립트 말고도 많은 프로그래밍언어에 내장되어있다.
,            Python에서도 JAVA도 가능하고, 오로지 class만 써야하는 언어도있다.
,
,            *** 새로운 의문.
,            User에서 Admin의 함수인 deleteWebsite를 호출할 수 있을까.
,            그것은 되지않는다.
,            admin 인스턴스만 sexyUser를 호출할 수 있다.
,
,            * admin인스턴스는 오로지 admin user이다.
,            sexyUser는 user의 인스턴스이며,
,            admin 인스턴스는 admin 인스턴스와 user 인스턴스를 합친것.
,
,         */
        class User {
            constructor({ username, lastName, email, password}) {
                this.username =  username;
                this.lastName = lastName;
                this.email = email;
                this.password = password;
            }
            sayHello() {
                console.log(`hello my name is ${this.username}`);
            }

            getProfile() {
                console.log(`${this.username}, ${this.email}, ${this.password}`)
            }
            updatePassword(newPassword, currentPassword) {
                if (currentPassword == this.password) {
                    this.password = newPassword
                } else {
                    console.log("can't change password")
                }
```

```
            }
        }

        const sexyUser = new User({
            username: "bibi",
            lastName: "bucky",
            email: "barnesquiat@gmail.com",
            password : 1111
        });
        console.log(sexyUser);

        class Admin extends User{
            constructor({ username, lastName, email, password, superAdmin, isActive}) {
                super({ username, lastName, email, password });
                this.superAdmin = superAdmin;
                this.isActive = isActive;
            }
            deleteWebsite() {
                console.log('Deleting the whole website')
            }
        }
        const admin = new Admin({
            username: "bibi",
            lastName: "bucky",
            email: "barnesquiat@gmail.com",
            password : 1111,
            superAdmin : true,
            isActive : false,
        })
        console.log(admin)


    </script>
</body>

</html>
```