

```

1  <!DOCTYPE html>
2  <html lang="en">
3
4  <head>
5      <meta charset="UTF-8">
6      <meta http-equiv="X-UA-Compatible" content="IE=edge">
7      <meta name="viewport" content="width=device-width, initial-scale=1.0">
8      <title>super</title>
9  </head>
10
11 <body>
12
13     <script>
14         /*
15         ,         ***** extend *****
16         ,
17         ,         이제 'admin'은 'user'의 extension이다.
18         ,         여기서 문제점은, 나는 더 많은것들을 추가하게 하고 만들고 싶다.
19         ,         'admin'은 'user'이지만 user 이상이기 때문.
20         ,
21         ,         예를들어 'super admin의 true/false를 생성해보자'
22         ,         constructor(superAdmin) {
23         ,             this.superadmin = superAdmin
24         ,         }
25         ,         admin안에 이런식으로 constructor를 정의해주면, 기존의 constructor를 잃어버린다.
26         ,         그래서 'admin'은 name부터 password까지가 뭔지 모르고있다.
27         ,
28         ,         먼저 user class를 리팩토링 먼저 할것이다.
29         ,         왜냐하면 여기서(인자 받는부분) 유저클래스가 값들을 많이 가질것이기 때문.
30         ,         options 오브젝트로 넣는것이 좋을듯하다.
31         ,
32         ,         const sexyUser = new User({
33         ,             username: "bibi",
34         ,             lasttName: "bucky",
35         ,             email: "barnesquiat@gmail.com",
36         ,             password : 1111
37         ,         });
38         ,         만약 내가 여러 arguments를 가지고 있다면 opttions오브젝트로 하는것이낫다
39         ,         어떤 옵션을 넘겨주는지 볼 수 있기 때문.
40         ,
41         ,         constructor({ username, lastName, email, password}) {
42         ,             this.username = name;
43         ,             this.lastName = lastName;
44         ,             this.email = email;
45         ,             this.password = password;
46         ,         }
47         ,         이런식으로 보내주어도 결과는 같다.
48         ,
49         ,         이전에 우리가 가졌던 문제점은 바로 admin을 만들고 싶었던것이다.
50         ,         user의 data와 몇가지 새로운 데이터를 포함해서.
51         ,         하지만 admin 안에서 새로운 constructor를 정의하니
52         ,         기존에 갖고있던것이 없어져서 오류가났다.
53         ,
54         ,         만약 우리가 constructor없이 admin을 만든다면 상관없다.
55         ,         왜냐하면 기존에 가지고있는것을 무시하지 않기때문.
56         ,         하지만 새로 만든다면 user constructor를 잃는다.
57         ,
58         ,         ***** super *****
59         ,         이럴때에 특별한 super함수를 호출하는것이다.
60         ,         이 함수는 classes안에서만 유효하다.
61         ,         super가 무엇을하느냐.

```

```

62 ,          super는 호출할것이다. constructor의 Base class(원시 클래스, 현재는 user)
63 ,
64 ,          class Admin extends User{
65 ,              constructor({ username, lastName, email, password, superAdmin,
isActive}) {
66 ,                  super({ username, lastName, email, password });
67 ,                  this.superAdmin = superAdmin;
68 ,                  this.isActive = isActive;
69 ,              }
70 ,              deleteWebsite() {
71 ,                  console.log('Deleting the whole website')
72 ,              }
73 ,          }
74 ,          const admin = new Admin({
75 ,              username: "bibi",
76 ,              lastName: "bucky",
77 ,              email: "barnesquiat@gmail.com",
78 ,              password : 1111,
79 ,              superAdmin : true,
80 ,              isActive : false,
81 ,          })
82 ,          console.log(admin)
83 ,
84 ,          우리는 이 admin에게 user의 옵션을 주어야하는데
85 ,          기존의 넘겨받던 arguments에 추가로 내가 몇가지 options들을 정의하고
86 ,          super에 base class의 constructor정보를 넘겨줄것이다.
87 ,          그러면 super 는 base class인 user의 constructor를 호출할것이다.
88 ,          그럼 기존에 갖고있던 username,lastName .. 을 넘겨받는다.
89 ,
90 ,          class는 자바스크립트 말고도 많은 프로그래밍언어에 내장되어있다.
91 ,          Python에서도 JAVA도 가능하고, 오로지 class만 써야하는 언어도있다.
92 ,
93 ,          *** 새로운 의문.
94 ,          User에서 Admin의 함수인 deleteWebsite를 호출할 수 있을까.
95 ,          그것은 되지않는다.
96 ,          admin 인스턴스만 sexyUser를 호출할 수 있다.
97 ,
98 ,          * admin인스턴스는 오로지 admin user이다.
99 ,          sexyUser는 user의 인스턴스이며,
100 ,          admin 인스턴스는 admin 인스턴스와 user 인스턴스를 합친것.
101 ,
102 ,          */
103 class User {
104     constructor({ username, lastName, email, password}) {
105         this.username = username;
106         this.lastName = lastName;
107         this.email = email;
108         this.password = password;
109     }
110     sayHello() {
111         console.log(`hello my name is ${this.username}`);
112     }
113
114     getProfile() {
115         console.log(`${this.username}, ${this.email}, ${this.password}`)
116     }
117     updatePassword(newPassword, currentPassword) {
118         if (currentPassword == this.password) {
119             this.password = newPassword
120         } else {
121             console.log("can't change password")
122         }

```

```
123     }
124   }
125
126   const sexyUser = new User({
127     username: "bibi",
128     lastName: "bucky",
129     email: "barnesquiat@gmail.com",
130     password : 1111
131   });
132   console.log(sexyUser);
133
134   class Admin extends User{
135     constructor({ username, lastName, email, password, superAdmin,
136 isActive}) {
137       super({ username, lastName, email, password });
138       this.superAdmin = superAdmin;
139       this.isActive = isActive;
140     }
141     deleteWebsite() {
142       console.log('Deleting the whole website')
143     }
144   }
145   const admin = new Admin({
146     username: "bibi",
147     lastName: "bucky",
148     email: "barnesquiat@gmail.com",
149     password : 1111,
150     superAdmin : true,
151     isActive : false,
152   });
153   console.log(admin)
154
155   </script>
156 </body>
157
158 </html>
```