```
<!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>Classes</title>
</head>

<body>

    <script>
        /*
,            ***** this ****
,            this는 기본적으로 클래스 안에서 볼 수 있고 클래스 그 자체를 가리킨다.
,            그래서 클래스 전체를 가리킨다고 볼 수 있는것.
,            내가 언제든 추가하고싶거나 클래스로부터 어떤것을 불러오고 싶을때 나는 this를 사용할것이다.
,
,            만약 클래스안에 두기만한다면 아주 쉬운개념일것이다.
,            하지만 funuction안의 this 그리고 어떤 일들에 대해 얘기할때는 어려울것이다.
,            내가 어떻게 나의 class와 function을 정의하느냐에 따라 달려있다.
,
,            * 이제부터 this는 클래스 그 자체를 가리킨다고 기억.
,            그래서 this.username과 name은 내가 무엇을 입력하든지 같다.
,
,            properties를 만들었다. 이것을 어떻게 만드는지 이해하는것은 중요하다.
,            또 임의로 만들수있다.
,            this.something = "hello something" 이런식으로.
,            constructor에서 argument로 받아서 불러올 필요도 없다.
,
,            ***** extending classes *****
,
,            updatePassword(newPassword, currentPassword) {
,                if (currentPassword == this.password) {
,                    this.password = newPassword
,                } else {
,                    console.log("can't change password")
,                }
,            }
,
,            기존 패스워드를 업데이트하는 방식으로 코드를 만들어보았다.
,            함수를 실행하면서 argument로 받아오는 값에 따라서 if문이 실행되도록.
,            이제 어떻게 class안에 data를 작성해야하는지 알게되었다.
,            클래스안에 내가 원하는 만큼 function을 추가하고 properties를 이용할수있다.
,
,            한개의 unit, 한개의 큰 object로 모든 함수를 클래스안에 담을 수 있다는 점에서 아주 멋지
다.
,            data를 불러올 수도, 바꿀 수도있다.
,
,            이제 이 user class를 extend해볼것이다.
,            예를들어 user class를 가지고 있고 어떤 부분을 약간 수정하고 싶다면 admin과 admin
part person을 위한 class를 만들어주어야한다.
,            admin person은 'sayHello'를 부를 수 있고, profile을 가지고있고, password를 업
데이트 할 수있다.
,            또한 더 많은것들을 'admin person'에 추가할 수 있다. (admin 만 할수있는 사이트 전체
삭제 이런것들)
,
,            바로 여기서 extend가 쓰이는것이다.
,            user 클래스에서 extend해서 new class를 만들것이다.
,
,            class Admin {
```

```
,                constructor(name, lastName, email, password) {
,                    this.username =  name;
,                    this.lastName = lastName;
,                    this.email = email;
,                    this.password = password;
,                    // this.something = "hello something"
,                }
,                sayHello() {
,                    // console.log(this) 클래스를 출력
,                    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")
,                    }
,                }
,            }
,            user class에서 추가하는 것이니까 복붙을 할것이다. 하지만 좋지않아보인다
,            왜냐하면 user와 admin의 이름이 같기 때문.
,            class Admin extends User{
,                그래서 이런식으로 extends를 사용해 'user'클래스에서 불러올것이다.
,            }
,            admin class는 deltewebsite라는 function을 가지고있다고 해보자.
,
,
,            class Admin extends User{
,                deleteWebsite() {
,                    console.log('Deleting the whole website')
,                }
,            }
,
,            const sexyAdmin = new Admin("bibi", "bucky", "barnesquiat@gmail.com",
1111)
,            sexyAdmin.deleteWebsite();
,            console.log(sexyAdmin.email)
,
,            new Admin 뒤에 인자를 넣어주지 않았을 경우에는 undefined가 뜰것이다.
,            이유는 constructor안에 아무것도 작성하지 않았기 때문, 기존에 썼던 ("bibi", ...)부분
을 붙여넣기 할것이다
,            왜냐하면 'admin'은 'user'이기 때문.
,            'admin'은 name, lastName, email, password을 가지고 있기 때문.
,
,
,            * 이 말은 이제 내가 'user' class인 'new admin'을 생성했고,
,            안에 있는것들이 user속으로 'admin'에 있는것들이 forwarding하고있는것이다.
,            그래서 sexyadmin.email을 부르는것과 sexyuser.email을 부르는것이 같게 된것.
,
,         */
        class User {
            constructor(name, lastName, email, password) {
                this.username =  name;
                this.lastName = lastName;
                this.email = email;
                this.password = password;
                // this.something = "hello something"
```

```
            }
            sayHello() {
                // console.log(this) 클래스를 출력
                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("bibi", "bucky", "barnesquiat@gmail.com", 1111);
        sexyUser.getProfile();
        sexyUser.updatePassword("hello", 1234)
        console.log(sexyUser.password)

        class Admin extends User{
            deleteWebsite() {
                console.log('Deleting the whole website')
            }
        }

        const sexyAdmin = new Admin("bibi", "bucky", "barnesquiat@gmail.com", 1111, true)
        sexyAdmin.deleteWebsite();
        console.log(sexyAdmin.email)

    </script>
</body>

</html>
```