Комментарии к коду

Жерносек Захар

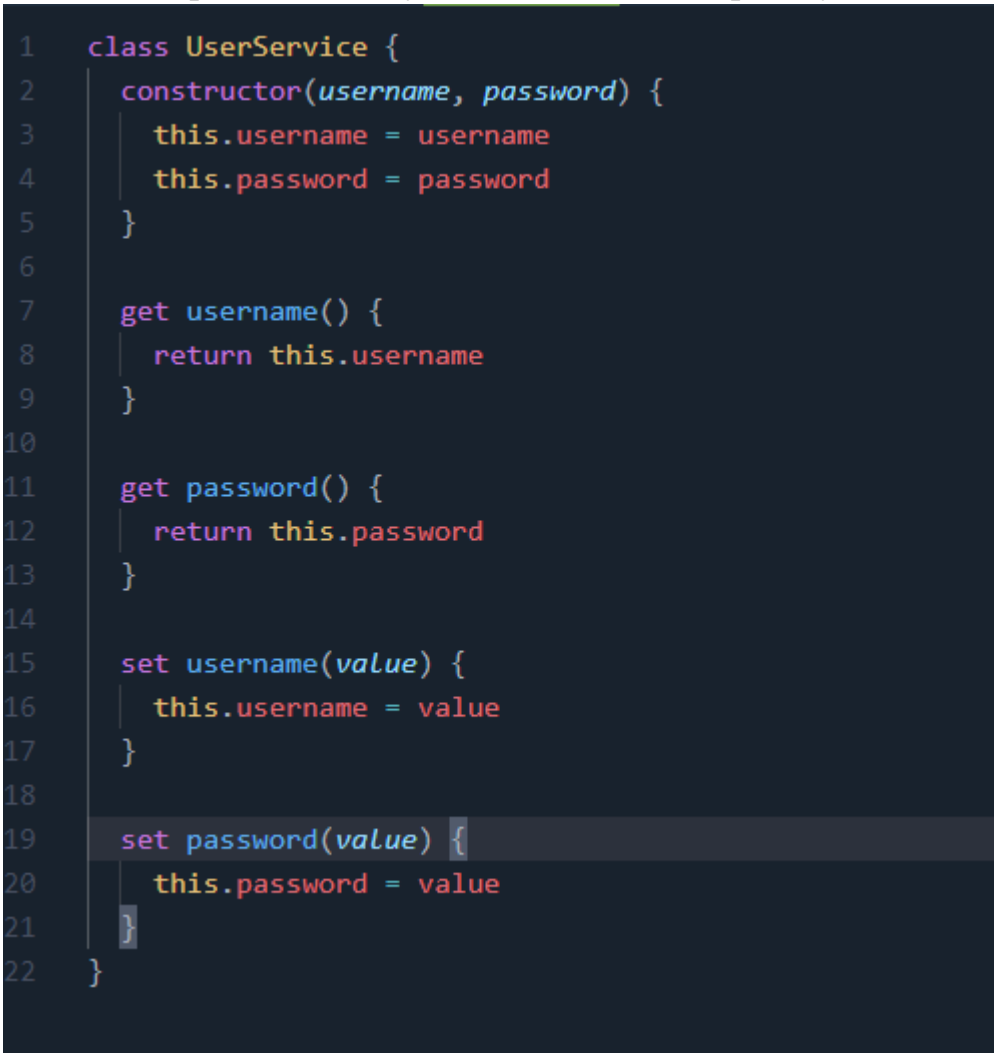
1Код на входе

```
class UserService {
    var username;
    var password;

    constructor(username, password) {
        this.username = username;
        this.password = password;
    }

    get username() {
        return UserService.username;
    }

    get password() {
        throw "You are not allowed to get password";
    }

    static authenticate_user() {
        let xhr = new XMLHttpRequest();
        xhr.open('GET', 'https://examples.com/api/user/authenticate?username=' +
            UserService.username + '&password=' + UserService.password, true);
        xhr.responseType = 'json';

        const result = false;

        xhr.onload = function() {
            if(xhr.status !== '200') {
                result = xhr.response;
            } else {
                result = true;
            }
        };

        return result;
    }
}

$('form #login').click(function () {
    var username = $('#username');
    var password = $('#password');

    var res = UserService(username, password).authenticate_user();

    if(res == true) {
        document.location.href = '/home';
    } else {
        alert(res.error);
    }
})
```

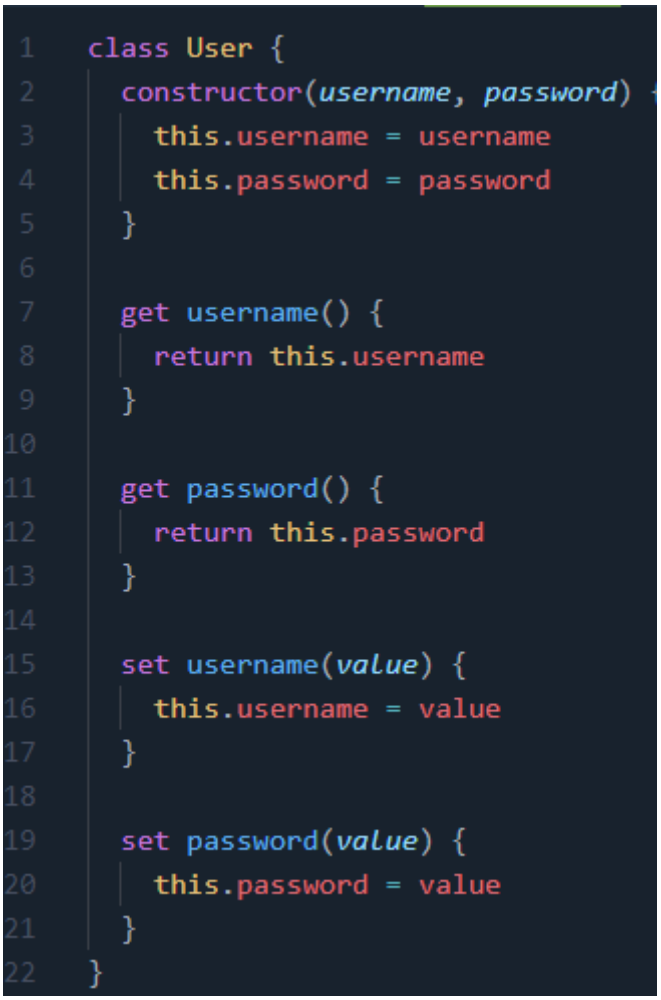
Проблемы

1. Неправильно объявлены поля класса
```
var username;
var password;
```
Необходимо было объявить так
```
class UserService {
  username;
  password;
```
, без ключевых слов var. Или мы можем вовсе их не объявлять, т.к. у нас дальше идёт constructor

2. В геттере username мы обращаемся к полю username как к статическому. Необходимо заменить UserService на this.
```
get username() {
    return UserService.username;
}
```
3. У нас есть геттеры, но мы не устанавливаем сеттеры. Лучше сделать так:



4. На данном этапе у нас возникает отдельная сущность, которую можно вынести из класса UserService
   Вынесем класс User:



```
class User {
  constructor(username, password) {
    this.username = username
    this.password = password
  }

  get username() {
    return this.username
  }

  get password() {
    return this.password
  }

  set username(value) {
    this.username = value
  }

  set password(value) {
    this.password = value
  }
}
```

5. XMLHttpRequest – устаревший метод, лучше воспользоваться axios или fetch. Также испоьзуем метод post, по протоколу HTTPS. Также метод лучше именовать как camelCase а не через _ .

```
async authenticateUser() {
  const url = `https://examples.com/api/user/authenticate`
  const headers = {
    "Content-Type": "application/json"
  }
  const body = JSON.stringify({
    username: this.user.username,
    password: this.user.password,
  })

  const response = await fetch(url, {
    method: "POST",
    headers,
    body
  })

  const result = response.ok ? true : await response.json()
  return result
}
}
```

6. Взаимодействие с DOM лучше написать так:

```
const loginForm = document.querySelector('#login')
loginForm.addEventListener("submit", async () => {
  const data = new FormData(loginForm)
  const username = data.get("username")
  const password = data.get("password")

  const user = new User(username, password)
  const userService = new UserService(user)
  const result = await userService.authenticate()

  if(result === true) {
    window.location.replace("/home")
  } else {
    alert(result.error)
  }
})
```

Здесь мы используем ванильный javascript вместо JQuery, Переходим с var на let и const, также я исправил семантические ошибки.

В ИТОГЕ

```
class User {
  constructor(username, password) {
    this.username = username
    this.password = password
  }

  get username() {
    return this.username
  }

  get password() {
    return this.password
  }

  set username(value) {
    this.username = value
  }

  set password(value) {
    this.password = value
  }
}
```

```
class UserService extends User {
  constructor(user) {
    this.user = user
  }

  async authenticateUser() {
    const url = `https://examples.com/api/user/authenticate`
    const headers = {
      "Content-Type": "application/json"
    }
    const body = JSON.stringify({
      username: this.user.username,
      password: this.user.password,
    })

    const response = await fetch(url, {
      method: "POST",
      headers,
      body
    })

    const result = response.ok ? true : await response.json()
    return result
  }
}

const user = new User("jolt", "12345")
console.log(user.username)
```

```
const loginForm = document.querySelector('#login')
loginForm.addEventListener("submit", async () => {
  const data = new FormData(loginForm)
  const username = data.get("username")
  const password = data.get("password")

  const user = new User(username, password)
  const userService = new UserService(user)
  const result = await userService.authenticate()

  if(result === true) {
    window.location.replace("/home")
  } else {
    alert(result.error)
  }
})
```