API Proxy Pattern: Reusable and Mockable API Calls

Allan A. Chua
https://www.pogsdotnet.com

GITHUB DEMO LINK

https://bit.ly/2HhiRVv

Slide Deck Link

https://bit.ly/2TH1mDM

Agenda

- What is Proxy Design Pattern
- Problems it solves
- How to implement
- How to improve
- How it makes testing easy
- The Future

What is Proxy Pattern

- Provide a surrogate or placeholder for another object to control access to it.
- Use an extra level of indirection to support distributed, controlled, or intelligent access.
- Add a wrapper and delegation to protect the real component from undue complexity.

How it Looks Like

How it Looks Like

users-proxy.js

```
1   import axios from "axios";
2
3   /**
4    * @class UsersProxy
5    * @description Proxy used for communicating with user endpoints.
6    * @author Allan A. Chua
7    * @version 1.0
8    * @since March 12, 2019
9    */
    Allan, 2 hours ago | 1 author (Allan)
10  export default class UsersProxy {
11    /**
12     * @async
13     * @function loadAllUsers
14     * @description Method used for pulling all user records.
15     *
16     * @returns {Promise<User[]>} Promise representing a set of users.
17     */
18    async loadAllUsers() {
19      try {
20        let { data } = await axios.get("http://localhost:3000/users");
21        Allan, 2 hours ago • -Implemented basic proxy demo
22        return data.users;
23      } catch (error) {
24        // TODO: Implement proper logging here.
25        return [];
26      }
27    }
28    /**
29     * @async
30     * @function getByID
31     * @description Method used for pulling user by ID.
32     *
33     * @param {number} id ID of user to retrieve.
34     *
35     * @returns {Promise<User>} Promise representing target user.
36     */
37    async getByID(id) {
38      try {
39        let { data } = await axios.get(`http://localhost:3000/users/${id}`);
40
41        return data.user;
42      } catch (error) {
43        // TODO: Implement proper logging here.
44        return [];
45      }
46    }
47  }
48
```

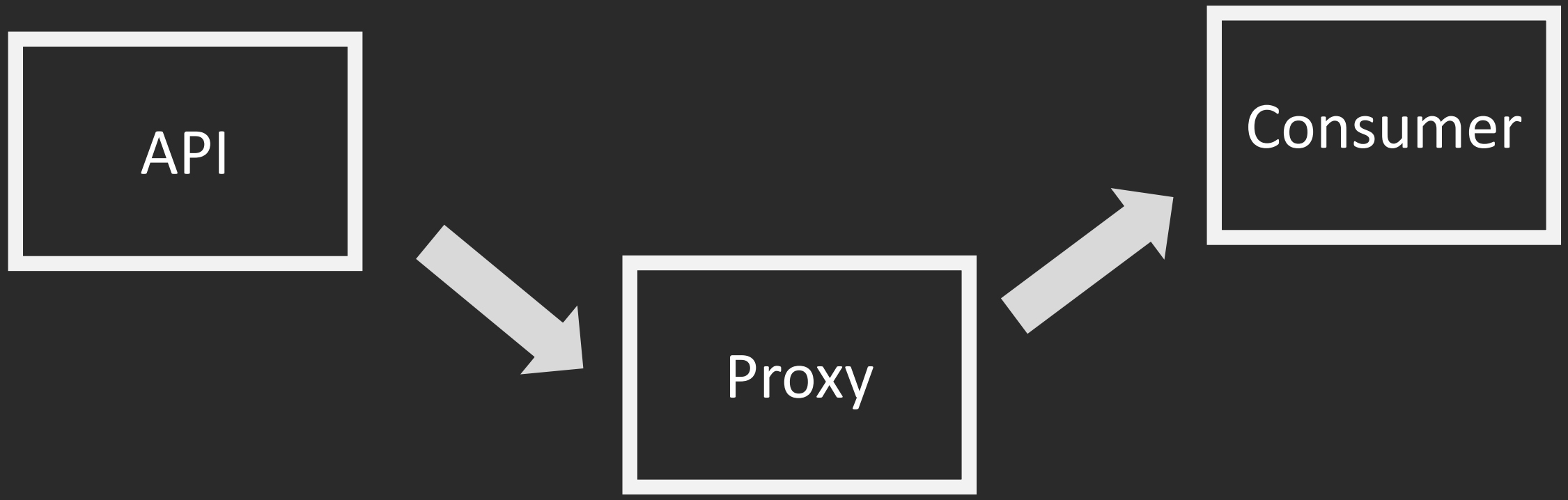
How it Looks Like

load-all-proxy.js - js-proxies - Visual Studio Code

load-all-proxy.js

```
Allan, 43 minutes ago | 1 author (Allan)
1  // import axios from "axios";
2  const axios = require("axios");
3
4  /**
5   * @class loadAllUsersProxy
6   * @description Proxy used for communicating with get all users endpoints.
7   * @author Allan A. Chua
8   * @version 1.0
9   * @since March 12, 2019
10  */
11 module.exports = {
12   /**
13    * @async
14    * @function loadAllUsers
15    * @description Method used for pulling all user records.
16    *
17    * @returns {Promise<User[]>} Promise representing a set of users.
18    */
19   async loadAllUsers() {
20     try {
21       let { data } = await axios.get("http://localhost:3000/users");
22
23       return data.users;
24     } catch (error) {
25       // TODO: Implement proper logging here.
26       return [];
27     }
28   }
29 };
30
```

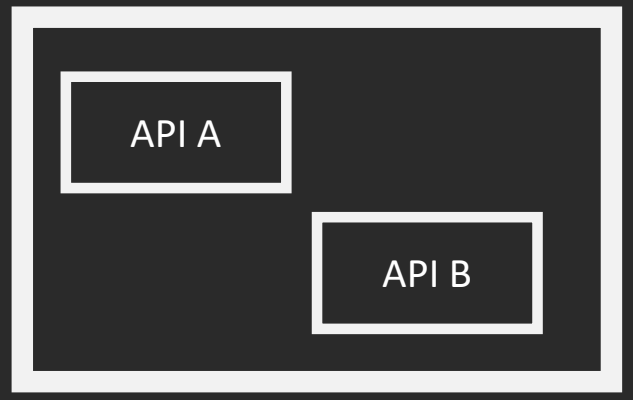
Why its perfect for API Calls



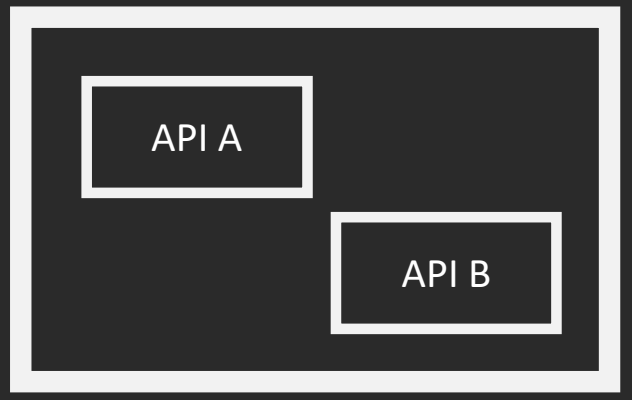
Encapsulates protocol and framework used to call API

Bad API Reuse

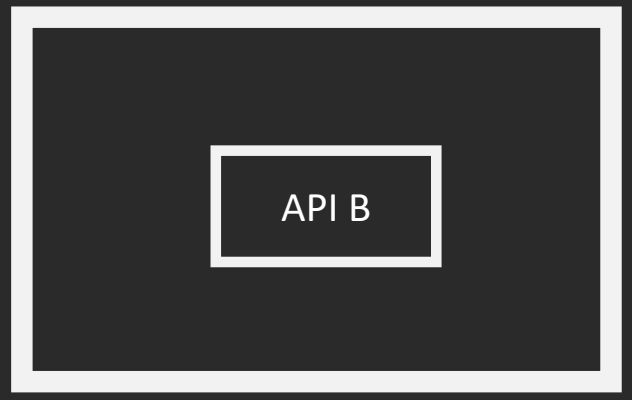
Consumer A



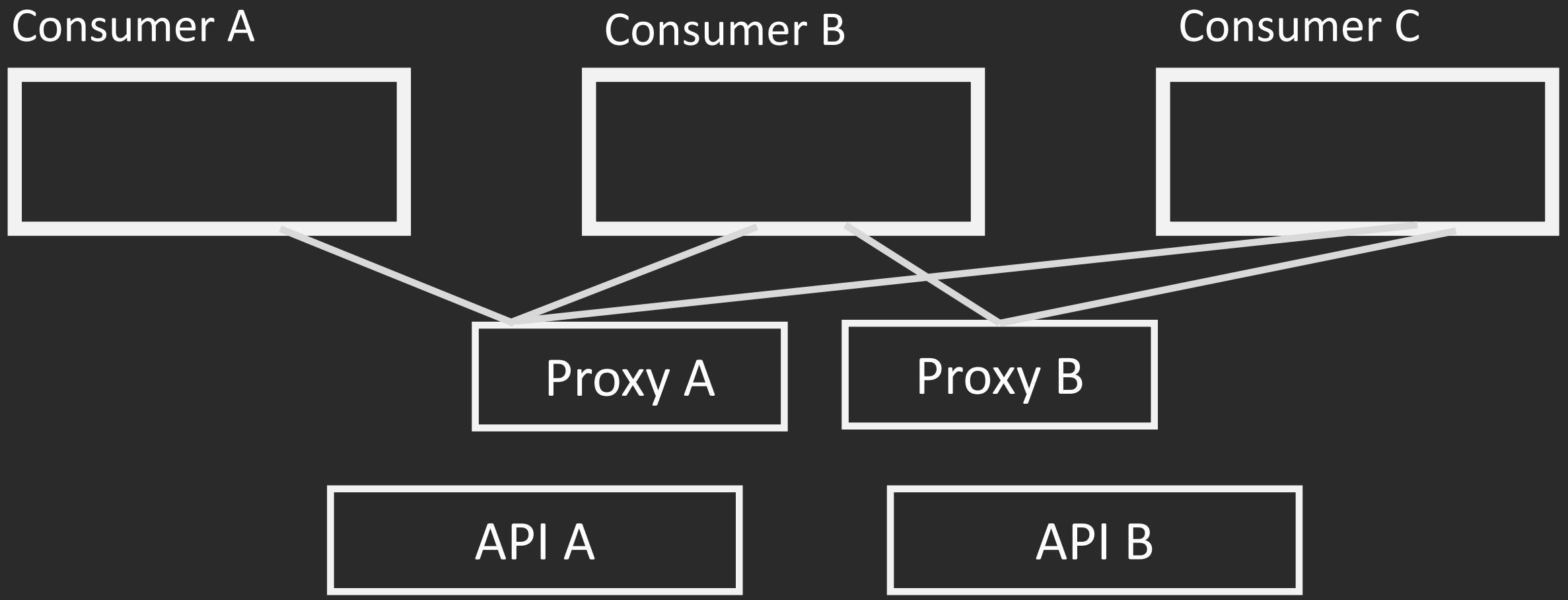
Consumer B



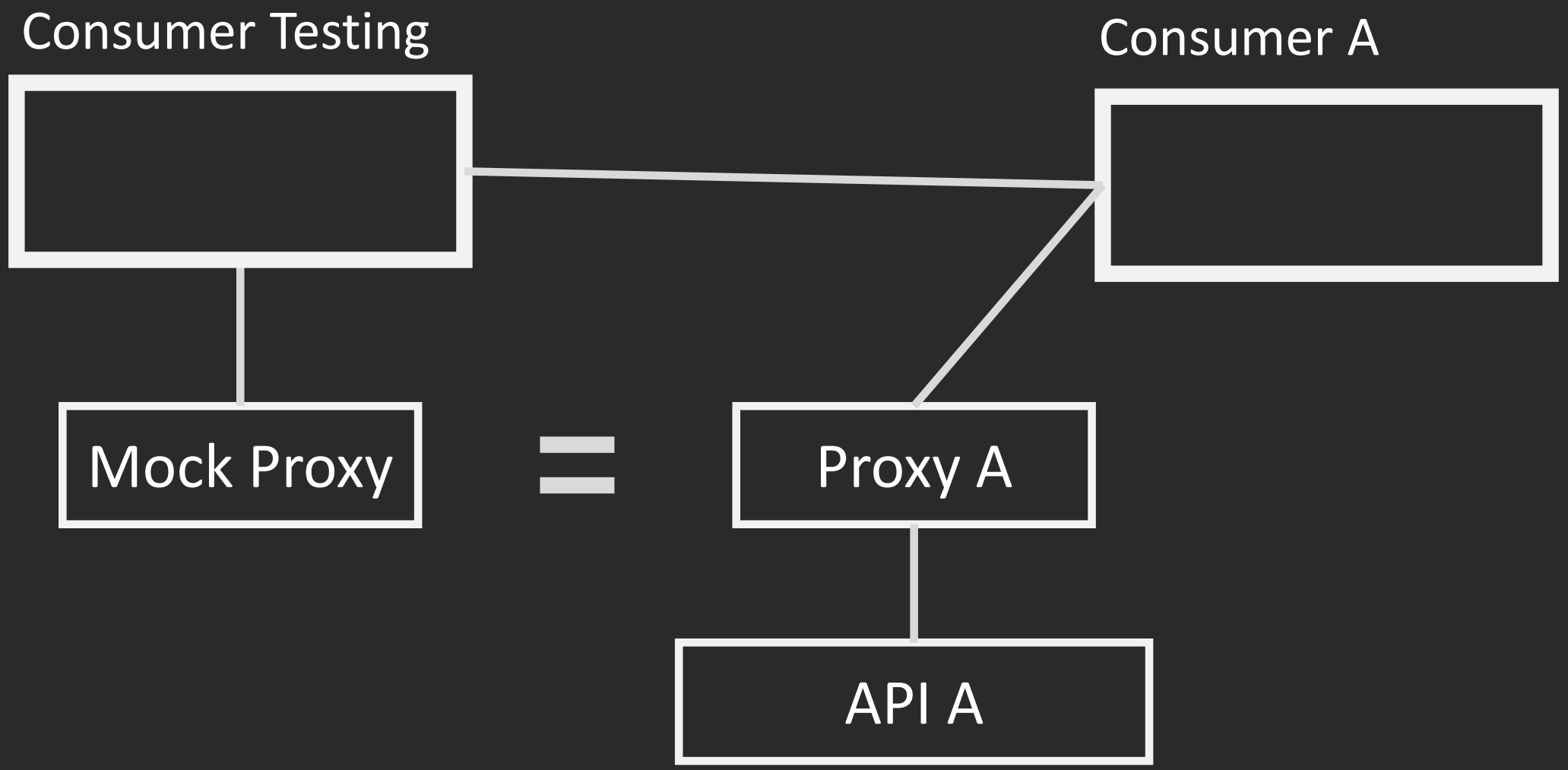
Consumer C



Good API Reuse



Easy and Predictable Testing



ES6 Proxies

```
let validator = {
  set: function(obj, prop, value) {
    if (prop === 'age') {
      if (!Number.isInteger(value)) {
        throw new TypeError('The age is not an integer');
      }
      if (value > 200) {
        throw new RangeError('The age seems invalid');
      }
    }

    // The default behavior to store the value
    obj[prop] = value;

    // Indicate success
    return true;
  }
};

let person = new Proxy({}, validator);

person.age = 100;
console.log(person.age); // 100
person.age = 'young'; // Throws an exception
person.age = 300; // Throws an exception
```


THANK YOU!
YOU ROCK!
memes.com