```
 1  /* ----------------------------------------------------------------
 2
 3  1. Example: Fetch API Data & Use REST Operator to Remove Password
 4
 5  ---------------------------------------------------------------- */
 6
 7  // Fake array of users (like an API response)
 8
 9  const fakeUsers = [
10
11      { id: 1, name: "Heet", email: "heet@example.com", password: "secret123" },
12
13      { id: 2, name: "Aisha", email: "aisha@example.com", password: "pass456" }
14  ];
15
16  /*
17
18  We want to display user data, BUT for security reasons, we should never display the password.
19
20  👉 Trick: Use object destructuring with the REST operator (...) to remove the `password` field while keeping the other
    properties.
21
22  */
23
24  const safeUsers = fakeUsers.map((user) => {
25
26      // Extract `password`, then collect the remaining properties into safeUser
27
28      const { password, ...safeUser } = user;
29
30      return safeUser; // return the "safe" user object without password
31  });
32
33  // Select the <div id="user-list"> from HTML to display users
34
35  const userList = document.getElementById("user-list");
36
37  // Loop through safeUsers and create HTML elements for each
```

```
38
39   safeUsers.forEach((user) => {
40
41       const div = document.createElement("div"); // create a new <div>
42
43       div.className = "user-card"; // add a class for styling
44
45       // Fill the div with user info (name + email only, password removed)
46
47       div.innerHTML = `<strong>Name:</strong> ${user.name}<br> <strong>Email:</strong> ${user.email}`;
48
49       // Finally, add this div inside the #user-list container
50
51       userList.appendChild(div);
52   });
53
54
55   /* ----------------------------------------------------------------
56
57   2. Example: Merge Default Settings with User Settings (Spread Operator)
58
59   ---------------------------------------------------------------- */
60
61   // Default website settings
62
63   const defaultSettings = { theme: "light", notifications: true, language: "en" };
64
65   // Suppose the user changed some settings
66
67   const userSettings = { theme: "dark", language: "fr" };
68
69   /*
70
71   👉 Spread operator (...) allows us to merge objects easily. If properties overlap, later values overwrite earlier ones. Here,
        userSettings will overwrite defaultSettings.
72
73   */
74
```

```
75   const mergedSettings = { ...defaultSettings, ...userSettings };

76

77   // Show the merged settings inside the <div id="settings"> in formatted JSON

78

79   document.getElementById("settings").textContent = JSON.stringify(mergedSettings, null, 2);

80

81   /*

82

83   ⚡ JSON.stringify(obj, null, 2)

84

85      - Converts object to JSON string

86

87      - `null` means no special filtering

88

89      - `2` means add indentation (pretty printing with 2 spaces)

90   */

91

92

93   /* ----------------------------------------------------------------

94

95   3. Example: Create Dynamic Form Inputs with Rest Operator

96

97   ---------------------------------------------------------------- */

98

99   // Get the form and the extra-fields container from HTML

100

101  const form = document.getElementById("userForm");

102

103  const extraFieldsContainer = document.getElementById("extra-fields");

104

105  // Add functionality: when "Add Extra Field" button is clicked

106

107  document.getElementById("addField").addEventListener("click", () => {

108

109      // Create a new <input>

110

111      const input = document.createElement("input");

112
```

```
113        input.type = "text"; // input type will be "text"

114

115        input.placeholder = "Extra Field"; // placeholder text

116

117        input.required = true; // make it mandatory

118

119        // Append the new input inside the "extra-fields" section of the form

120

121        extraFieldsContainer.appendChild(input);

122

123        // Optional: add line break after each new input for clarity

124

125        extraFieldsContainer.appendChild(document.createElement("br"));

126  });

127

128  // When the form is submitted

129

130  form.addEventListener("submit", (event) => {

131

132        event.preventDefault(); // stop page from refreshing

133

134        // Get ALL inputs inside the form (including extra ones)

135

136        const inputs = Array.from(form.querySelectorAll("input"));

137

138        /*

139

140        Use destructuring with rest operator:

141

142        - First two inputs → firstName, lastName

143

144        - The remaining inputs → extraFields (as an array)

145

146        */

147

148        const [firstName, lastName, ...extraFields] = inputs.map((input) => input.value);

149

150        // Show the collected data in an alert box
```

```
151
152      alert(`First Name: ${firstName}\nLast Name: ${lastName}\nExtra Fields:\n${extraFields.join("\n")}`);
153
154      // Reset the form (clears inputs)
155
156      form.reset();
157  });
```