🎨 Data Cleaning & Validation Solutions

📧 Q1. Email Cleaning & Validation

Problem Statement: A user enters their email with extra spaces and in uppercase: `"   USER123@GMAIL.com   "`

Requirements: 👉 Write a program that:

1. Removes extra spaces.
2. Converts the email to lowercase.
3. Checks if the email contains `"@"`.
4. Checks if the email ends with `".com"`. Finally, print the cleaned email and the results.

Input Data:

```
"   USER123@GMAIL.com   "
```

Solution:

```javascript
let email = "   USER123@GMAIL.com   ";
let cleanedEmail = email.trim().toLowerCase();
console.log("Cleaned Email:", cleanedEmail);
console.log("Includes '@'? :", cleanedEmail.includes("@"));
console.log("Ends with '.com'? :", cleanedEmail.endsWith(".com"));
```

📤 Output:

```
Cleaned Email: user123@gmail.com
Includes '@'? : true
Ends with '.com'? : true
```

📱 Q2. Phone Number Normalization

Problem Statement: A customer provides their phone number with spaces and dashes: `" +91-98765 43210 "`

Requirements: 👉 Write a program that:

1. Removes extra spaces.
2. Removes spaces and dashes.
3. Checks if the phone number starts with `+91`. Finally, print the cleaned phone number and the result.

Input Data:

```
"  +91-98765 43210  "
```

Solution:

```javascript
let phone = "  +91-98765 43210  ";
let cleanedPhone = phone.trim().replaceAll(" ", "").replaceAll("-", "");
console.log("Formatted Phone:", cleanedPhone);
console.log("Starts with +91? :", cleanedPhone.startsWith("+91"));
```

📤 **Output:**

```
Formatted Phone: +919876543210
Starts with +91? : true
```

💳 Q3. Credit Card Masking

Problem Statement: A website wants to hide sensitive credit card numbers for security. Given: `"1234 5678 9012 3456"`

Requirements: 👉 Write a program that:

1. Extracts only the last 4 digits.
2. Replaces the rest of the digits with `"*"` while keeping the length same. Finally, print the masked card number.

Input Data:

```
"1234 5678 9012 3456"
```

Solution:

```javascript
```

```javascript
let card = "1234 5678 9012 3456";
let last4 = card.slice(-4);
let masked = last4.padStart(card.length, "*");
console.log("Masked Card:", masked);
```

📤 **Output:**

```
Masked Card: ***************3456
```

🏷️ Q4. Product Code Normalization

Problem Statement: An e-commerce system stores product codes but they may have inconsistent formatting. Example: `" ab-123 xy "`

Requirements: 👉 Write a program that:

1. Removes extra spaces.
2. Converts all letters to uppercase.
3. Removes internal spaces. Finally, print the normalized product code.

Input Data:

```
" ab-123 xy "
```

Solution:

```javascript
let product = " ab-123 xy ";
let cleanedProduct = product.trim().toUpperCase().replaceAll(" ", "");
console.log("Normalized Product Code:", cleanedProduct);
```

📤 **Output:**

```
Normalized Product Code: AB-123XY
```

🔐 Q5. Secure URL Check

Problem Statement: A website receives the following URL: `" http://example.com "`

Requirements: 👉 Write a program that:

1. Removes extra spaces.
2. Checks if the URL starts with `"https://"`.
    - If yes → Print `"Secure URL"`.
    - If no → Print `"Warning: URL is not secure!"`.

Solution:

```javascript
let url = "   http://example.com   ";
let cleanedUrl = url.trim();
if (!cleanedUrl.startsWith("https://")) {
  console.log("Warning: URL is not secure!");
} else {
  console.log("Secure URL:", cleanedUrl);
}
```

📤 **Output:**

```
Warning: URL is not secure!
```

📚 Summary of Methods Used

String Cleaning Methods:

- **`trim()`** - Removes leading and trailing whitespace
- **`toLowerCase()`** - Converts string to lowercase
- **`toUpperCase()`** - Converts string to uppercase
- **`replaceAll()`** - Replaces all occurrences of a substring

String Validation Methods:

- **`includes()`** - Checks if string contains a substring
- **`startsWith()`** - Checks if string starts with specified characters
- **`endsWith()`** - Checks if string ends with specified characters

String Manipulation Methods:

- **`slice()`** - Extracts a section of string
- **`padStart()`** - Pads string from the beginning with specified characters

---

✨ Key Learning Points

1. **Data Sanitization** is crucial for user input validation
2. **Method Chaining** allows for concise and readable code
3. **Security Practices** like masking sensitive information protect user data
4. **Consistent Formatting** improves data quality and system reliability
5. **Validation Checks** ensure data meets expected criteria before processing

---

🎉 All Solutions Successfully Implemented! 🎉