JavaScript Practical Regex

Coding Exercise Challenge

JavaScript Practical Regex: 10 Coding Questions with Answers

Regular expressions (regex) in JavaScript provide a powerful way to process text. By using patterns, you can perform complex text searches, replacements, and validations. Below are 10 coding questions covering various practical applications of regex in JavaScript.

Delving into the world of regular expressions (regex) can unlock a treasure trove of solutions for common text processing challenges. From validating email addresses to formatting dates, regex in JavaScript is an indispensable tool for efficient coding. We've explored ten practical regex applications, providing insights into how they can streamline your development process.

🔍 Key Regex Applications:

- Email Validation: Implement precise validation of email formats.
- Hashtag Extraction: Seamlessly extract hashtags from social media content.
- Password Strength Check: Enforce strong password policies in your applications.
- Whitespace Management: Clean up text inputs by replacing multiple spaces with a single one.
- Number Extraction: Quickly retrieve numerical data from mixed text.
- Date Formatting: Convert date formats for better user readability.
- HTML Tag Removal: Strip HTML tags from content for text processing.
- URL Identification: Detect and process URLs in large blocks of text.
- Hex Color Validation: Ensure user-inputted hex color codes are valid.

🖥️ Mastering regex can dramatically enhance your coding capabilities, making your scripts more robust, versatile, and maintainable.

Question 1: Validating Email Addresses

Q: How do you use regex to validate email addresses?

```
function validateEmail(email) {
  const regex = /^\w+([.-]?\w+)*@\w+([.-]?\w+)*(\.\w{2,3})+$/;
```

```
    return regex.test(email);
}
```

Explanation:

This regex checks for a general pattern in email addresses: characters before and after an '@' symbol and a domain at the end. It's important to note that email regex can get complex due to the vast variety of valid email formats.

Question 2: Extracting Hashtags from Text

Q: How do you extract hashtags from a string?

```
function extractHashtags(text) {
    const regex = /#\w+/g;
    return text.match(regex);
}
```

Explanation:

This regex looks for words preceded by the '#' symbol. The \w+ matches one or more word characters following the '#'.

Question 3: Checking for Palindromes

Q: How do you use regex to check if a string is a palindrome?

```
function isPalindrome(str) {
    str = str.toLowerCase().replace(/[\W_]/g, '');
    return str === str.split('').reverse().join('');
}
```

Explanation:

First, non-word characters and underscores are removed and the string is converted to lowercase. Then, it checks if the string is the same forwards and backwards.

Question 4: Replacing Multiple Spaces with a Single Space

Q: How do you replace multiple spaces in a string with a single space?

```
function replaceMultipleSpaces(text) {
  return text.replace(/\s+/g, ' ');
}
```

Explanation:

The regex \s+ matches one or more whitespace characters. The replace function replaces these with a single space.

Question 5: Extracting Numbers from a String

Q: How do you extract all numbers from a given string?

```
function extractNumbers(str) {
  return str.match(/\d+/g).map(Number);
}
```

Explanation:

The regex \d+ matches one or more digits. match returns an array of all digit sequences found, which is then converted to an array of numbers.

Question 6: Validating a Password Strength

Q: How can regex be used to validate strong passwords (minimum 8 characters, at least one letter, one number, and one special character)?

```
function validatePassword(password) {
  const regex =
/^(?=.*[A-Za-z])(?=.*\d)(?=.*[@$!%*#?&])[A-Za-z\d@$!%*#?&]{8,}$/;
  return regex.test(password);
}
```

Explanation:

This regex uses lookaheads to ensure that the password contains at least one letter ([A-Za-z]), one number (\d), and one special character ([@$!%*#?&]). The {8,} ensures the password is at least eight characters long.

Question 7: Formatting a Date String

Q: How do you use regex to format a date string from "YYYY-MM-DD" to "DD-MM-YYYY"?

```
function formatDate(date) {
  return date.replace(/^(\d{4})-(\d{2})-(\d{2})$/, '$3-$2-$1');
}
```

Explanation:

The regex captures three groups: year, month, and day. The replace function rearranges these groups into the desired format.

Question 8: Removing HTML Tags from a String

Q: How can you strip HTML tags from a string using regex?

```
function removeHtmlTags(str) {
   return str.replace(/<[^>]*>/g, '');
}
```

Explanation:

This regex matches anything that looks like an HTML tag (<[^>]*>) and replaces it with an empty string, effectively removing it.

Question 9: Finding URLs in Text

Q: How do you identify URLs within a block of text?

```
function findUrls(text) {
   const regex = /https?:\/\/[^\s]+/g;
   return text.match(regex);
}
```

Explanation:

The regex looks for substrings that start with http or https, followed by :// and then any non-whitespace characters, capturing typical URL patterns.

Question 10: Validating Hex Color Codes

Q: How do you validate hex color codes?

```
function isValidHexColor(code) {
  const regex = /^#?([a-f0-9]{6}|[a-f0-9]{3})$/i;
  return regex.test(code);
}
```

Explanation:

This regex checks for strings that start with an optional #, followed by either 3 or 6 hexadecimal characters. The i flag makes the regex case-insensitive, allowing for both lowercase and uppercase letters.