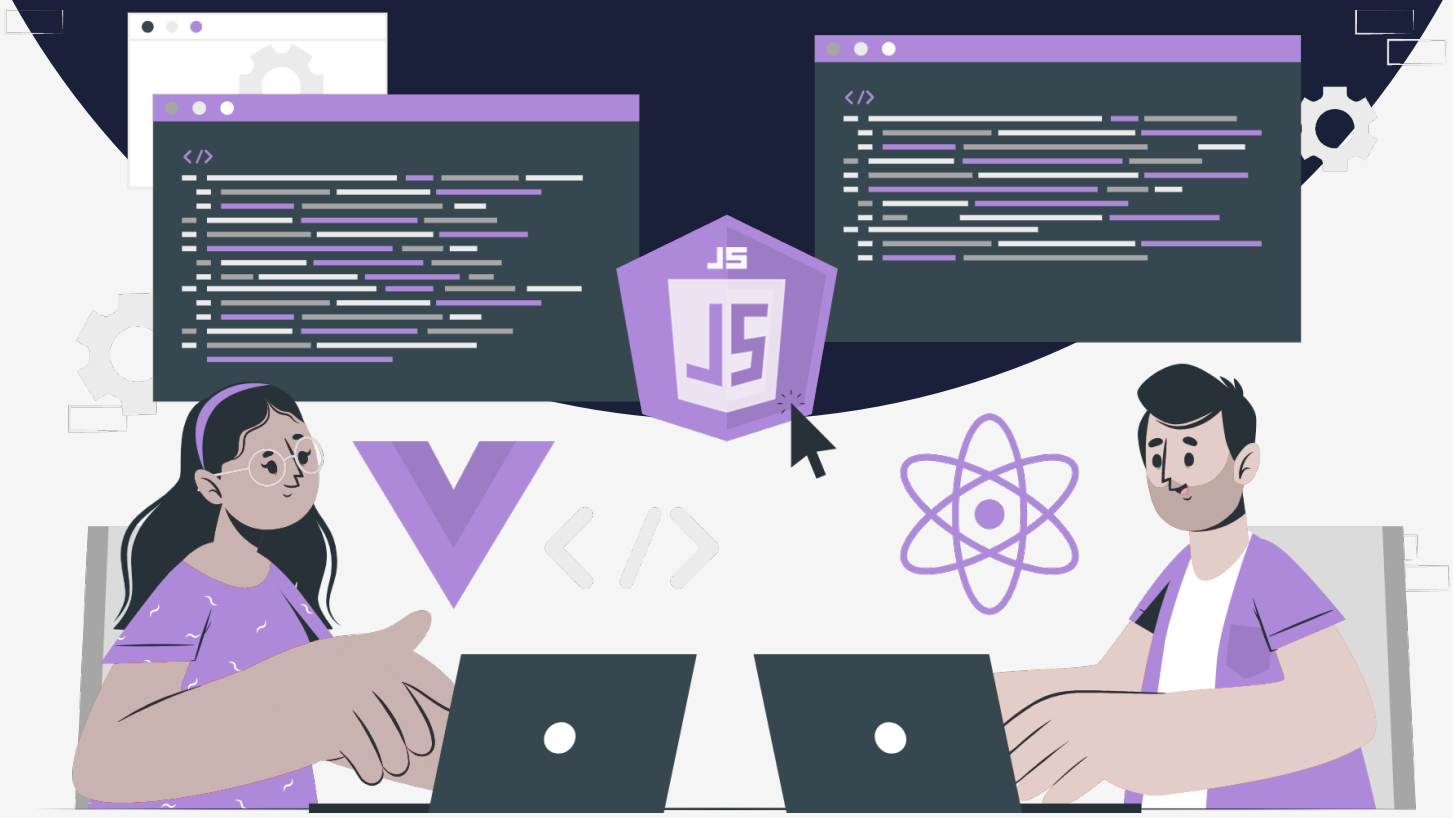
Lesson:

What is regex and Different Character in Regex and importance



PW SKILLS

Topics Covered

1. Introduction.
2. Use cases of regex.
3. The basic structure of regex.
4. Types of regex characters.

Regular expressions which are often referred to as regex or regexp are a powerful tool that helps developers work with strings or text. They are popularly used as a pattern-matching tool which is very much helpful in searching and manipulating specific strings in a text.

Regex is so powerful that we can use regex to match specific characters, words or even the entire string of a text document. They are most commonly used in text editors, and programming languages for tasks such as search, replace, and data validation.

Regex patterns are made up of a combination of characters and special symbols. The most basic regex pattern is a simple string of characters. For example, the pattern "hello" would match the string "hello" in a text document. We will look into many of such patterns in further lectures.

Regex also supports the use of character classes, which we will be looking further, it allows you to match a specific set of characters. For example, the character class "[a-z]" will match any lowercase letter.

Additionally, regex provides a number of special character sequences, such as \d to match any digit, \s to match any whitespace, and \w to match any word character.

Some of the most common use cases of a regex are:

- Search and replace: You can use regex to search for a specific pattern in a text document and replace it with another pattern. This is often used in text editors and programming languages to make changes to a document.

- Data validation: Regex can be used to ensure that a specific string of text conforms to a certain pattern, such as an email address or phone number. This can be used to check user input on a website or in a program to make sure that it is valid.

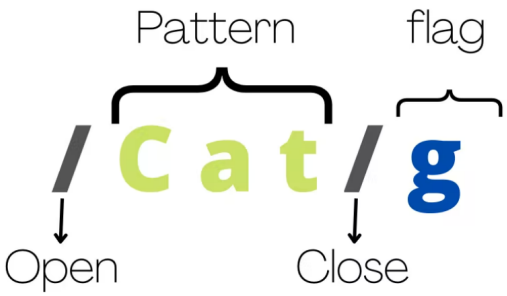
- Data Cleaning: Regex can be used to clean up messy data, such as removing unwanted characters or converting text to a consistent format.

By now you must have understood that regex is not limited to any particular programming language and field. Most of the languages have regex as their default property while some require some third-party packages to be installed, and there are many online tools to try regex before implementing on into the code.

We will look into the basic structure of regex in this lecture and look into the implementation and in-depth understanding of the parts in upcoming lectures.

The basic structure of regex has the following parts:

1. An opening tag.
2. A closing tag.
3. The pattern.
4. The flag.



An opening tag is the character or sequence of characters that indicate the start of a regex pattern. In javascript, the opening tag is a forward slash (/) or new RegExp(.

The closing tag indicates the end of a regex pattern. In javascript, the opening tag is a forward slash (/).

The pattern is the actual regular expression that you want to match. It can be a combination of characters or special characters.

The flags are optional and are used to modify the behavior of the regular expression.

The different characters in the regex can be mainly classified into

1. Flags.
2. Character classes.
3. Characters.
4. Anchors.
5. Quantifiers.

Flags:

Regex flags are optional options or modifications that can be added to a regex pattern to control its behavior.

Some of the common flags in the regex are

- g : This flag stands for global matching. When this flag is used it finds all matches in the input, not just the first.

- i : This flag stands for "case-insensitive". When this flag is used, the pattern matching becomes case-insensitive. It will match both uppercase and lowercase letters.

- m : This flag stands for "multiline". When this flag is used, the pattern matching treats the input string as multiple lines, rather than a single line.

- s : This flag stands for "single-line". When this flag is used, the pattern matching treats the input string as a single line, even if it contains newline characters.

- u : This flag stands for "Unicode". When this flag is used, the pattern matching operates on Unicode characters, rather than bytes or ASCII characters.

- x : This flag stands for "extended". When this flag is used, the pattern matching allows you to add whitespace and comments to the pattern for readability.

We will look into how to add whitespaces and comments in later sections.

Character Classes:

A character class matches any one of a given set of characters. It is used to match the basic elements of a language like a letter, a digit, a space, a symbol, etc.

- "\d" - matches any digit (0-9)
- "\w" - matches any word character (letters, digits, and underscore)
- "\s" - matches any whitespace character (space, tab, newline, etc.)
- "\D" - matches any character that is not a digit
- "\W" - matches any character that is not a word character
- "\S" - matches any character that is not a whitespace character

Characters:

- "[abc]" - matches any of the characters "a", "b", or "c"
- "[^abc]" - matches any character except "a", "b", or "c"
- "[a-z]" - matches any lowercase letter
- "[A-Z]" - matches any uppercase letter
- "[0-9]" - matches any digit
- “[set_of_characters]” – Matches any single character in set_of_characters. By default, the match is case-sensitive.
- \ - Escapes a special character.
- . - Wildcard character. It is used to match anything except line breaks.
- ^ - Negation or negative character. It will look for a pattern not equal to the mentioned pattern. It works inside character sets.

Anchors:

Anchors in regex are special characters that match specific positions within a string, rather than matching specific characters.

Some of the commonly used anchors are:

- "^" (caret) - matches the start of a string or line
- "$" (dollar) - matches the end of a string or line

Quantifiers:

Quantifiers are the repeaters in regex. These are used to specify the number of times a character, character class, or group should be repeated in a match.

- "*" (asterisk) - matches 0 or more occurrences of the preceding character or group
- "+" (plus) - matches 1 or more occurrences of the preceding character or group
- "?" (question mark) - matches 0 or 1 occurrence of the preceding character or group
- “{}” (curly brackets) - matches exactly the number specified within brackets occurrences of the preceding character or group.