What is RegEx?

A regular expression, also known as regex, is a pattern that represents a collection of strings that match the pattern.

Example: Regular expression for an email Address:
^ ([a-zA-Z0-9_\-\.]+) @ ([a-zA-Z0-9_\-\.]+)\. ([a-zA-Z]{2,5})$

Here are some of the most common uses:

- **Search and Replace:** Regex is often used to search for specific patterns in text and replace them with different text. This is useful in text editors, word processors, and programming languages.
- **Validation:** Regex is frequently used to validate input data, such as checking if an email address, phone number, or password meets certain criteria (e.g., length, character types).
- **Security:** Regex is employed in security contexts to detect patterns indicative of malicious activity, such as SQL injection attempts or other forms of attack.

Verification is Checking correctness against specifications.

Validation is Checking suitability for intended use.

Use Cases

Regular expressions have applications in a wide range of areas. We list some of the most popular ones below:

- email validation
- password validation
- date validation
- phone number validation
- search and replace in text editors

Online Regex Tools: Regex101 , RegExr etc.

Regex flags:

Characters	Meaning
g	Performs a global match, finding all matches rather than just the first.
i	Makes matches case-insensitive. Matches both uppercase and lowercase.
m	Performs multiline matches. (Changes behavior of ^,$)
s	The dot (.) matches all possible characters.
u	Enables Unicode support.
y	Matches are sticky, looking only at exact position in the text.

Character Classes: Character classes specify a given type of character to match.

Characters	Meaning
.	Matches any character except new line. When s flag set, it also matches line terminators.
\d	Matches any digit (Arabic numeral).
\D	Matches any character that is not a digit (Arabic numeral).
\w	Matches any alphanumeric character from Latin alphabet, including underscore.
\W	Matches any character that is not an alphanumeric character from Latin alphabet or underscore.
\s	Matches any whitespace character (space, tab, newline, non-breaking space, and similar).
\S	Matches any character that isn't a whitespace character.
\t	Matches a horizontal tab.
\r	Matches a carriage return.
\n	Matches a newline.
\v	Matches a vertical tab.
[abc]	Matches any one of the characters "a", "b", or "c".
[0-9]	Matches any digit.
[^abc]	Matches any character except "a", "b", or "c".

Characters	Description
[a-z]	matches any lower case letter
[A-Z]	matches any upper case letter
[0-9]	matches any digit
[a-zA-Z]	matches any lower/upper case letter
[a-zA-Z0-9]	matches any lower/upper case letter or digit
[^0-9]	matches everything except digits

Metacharacters: Characters that have special meanings, such as:

. (dot)	Matches any single character except a newline.
^	Matches the start of a string.
$	Matches the end of a string.
*	Matches 0 or more of the preceding element.
+	Matches 1 or more of the preceding element.
?	Matches 0 or 1 of the preceding element.
\	Escapes a metacharacter.

Anchors	Description
^r	matches any string that begins with r
r$	matches any string that ends with r
^r$	matches any string that starts and ends with r
r	matches any string that has r in it

Quantifiers: Specify how many times an element should appear. For example:

{n}: Exactly n times.

{n,}: At least n times.

{n,m}: Between n and m times.

Quantifier	Description
.	matches everything except a new line
?	the item to its left is optional and is matched at most once
*	the item to its left will be matched zero or more times
+	the item to its left is matched one or more times
{n}	the item to its left is matched exactly n times
{n, }	the item to its left is matched n or more times
{n, m}	the item to its left is matched at least n times but not more than m times

Groups and Ranges: Use parentheses to group parts of the pattern and capture matches. For example:

(abc): Matches "abc" and captures it.

OR	Description
a(b|c)	matches any string that has a followed by b or c
a[bc]	

Regex practice pattern:

Functions used in JavaScript:

- match(regex)= Returns an array containing all of the matches, including capturing groups, or null if no match is found,
- search(regex)= Tests for a match in a string. It returns the index of the match, or -1 if the search fails,
- replace(regex,'replace')= Executes a search for a match in a string, and replaces the matched substring with a replacement substring,
- test(text) =Tests for a match in a string. It returns true or false.

Structured Query Language (SQL)

Structured query language (SQL) is a programming language for storing and processing information in a relational database.

What is SQL injection (SQLi)?

SQL injection, also known as SQLI, is a common attack vector that uses malicious SQL code for backend database manipulation to access information that was not intended to be displayed.

What is the impact of a successful SQL injection attack?

A successful SQL injection attack can result in unauthorized access to sensitive data, such as:

- Passwords.
- Credit card details.
- Personal user information.

SQL Injection Types

There are different types of SQL injection attacks:

1. Error-based SQL Injection

In error-based SQL injection, the attacker tries to insert a malicious query with the goal of receiving an error message that provides sensitive information about the database.

Attackers exploit error messages generated by the web application by analyzing error messages to gain access to confidential data or modify the database.

2. Blind SQL Injection

Blind SQL injection is a type of SQL injection where the attacker does not receive an obvious response from the attacked database.

By analyzing the application's behavior, attackers can determine the success of the query.

3. Out-of-band SQL Injection

Uses a different channel to communicate with the database.

Methods to Prevent SQL Injection Attacks

The five key methods to prevent SQL injection attacks include:

1. **Filter database inputs:** Detect and filter out malicious code from user inputs.
2. **Restrict database access:** Prevent unauthorized data access, exfiltration, or deletion through access control restrictions.
3. **Maintain applications and databases:** Keep databases fully patched and updated. Upgrade when possible.

4. **Monitor application and database inputs and communications:** Monitor communication to detect and block malicious SQLi attempts.