**[Patterns and Flags](https://classes.capaciti.org.za/courses/course-v1:CITI+CAP004+2021/course/" \l "block-v1:CITI+CAP004+2021+type@sequential+block@3ff3b5623acc48bfabe29bc3ebd802fb)**

Patterns and flags are terms related to regular expressions, which are patterns that provide a powerful way to search and replace in text.

A regular expression consists of a pattern and optional flags. The pattern is a sequence of characters that defines the search criteria, while the flags are modifiers that affect the search behavior. There are two syntaxes that can be used to create a regular expression object in JavaScript:

* The long syntax: regexp = new RegExp("pattern", "flags");
* The short syntax, using slashes /: regexp = /pattern/; // no flags or regexp = /pattern/gmi; // with flags g, m and i

The slashes /.../ tell JavaScript that we are creating a regular expression. They play the same role as quotes for strings.

**To search for all matches:**

Use regexp g flag and:

* + Get a flat array of matches – str.match(reg)
  + Get an array or matches with details – str.matchAll(reg).

**To search for the first match only:**

* + Get the full first match – str.match(reg) (without g flag).
  + Get the string position of the first match – str.search(reg).
  + Check if there’s a match – regexp.test(str).
  + Find the match from the given position – regexp.exec(str) (set regexp.lastIndex to position).

**To replace all matches:**

* + Replace with another string or a function result – str.replace(reg, str|func)

**To split the string by a separator:**

str.split(str|reg)

Now you can continue reading this chapter to get the details about every method… But if you’re reading for the first time, then you probably want to know more about regexps. So you can move to the next chapter, and then return here if something about a method is unclear.

[**str.search(reg)**](https://javascript.info/regexp-methods#str-search-reg).

1. **Patterns**:
   * **Patterns** are the core of regular expressions. They are sequences of characters that define the search criteria you want to apply to a string. Patterns can include regular characters (letters, digits, symbols) as well as special metacharacters with specific meanings. For example:
     + **abc**: A simple pattern that matches the exact characters "abc."
     + **\d**: A metacharacter that matches any digit (0-9).
     + **.**: A metacharacter that matches any character except a newline.
2. **Flags**:
   * **Flags** are optional parameters or settings that modify how a regular expression behaves when applied to a string. Flags are usually placed after the pattern and are preceded by a forward slash **/** in JavaScript and many other programming languages. Common flags include:
     + **/i**: Case-insensitive matching, so that uppercase and lowercase characters are treated as equal.
     + **/g**: Global matching, which finds all occurrences of the pattern in the input string, rather than just the first one.
     + **/m**: Multiline matching, which allows the **^** and **$** metacharacters to match the start and end of each line within a multiline string.

Here's an example of a regular expression pattern with flags in JavaScript:

const pattern = /abc/gi; // Pattern to match "abc" case-insensitively and globally

const input = "Abc ABC abc XYZ";

const matches = input.match(pattern);

console.log(matches); // Output: [ 'Abc', 'ABC', 'abc' ]