REGEX

FUNDAMENTALS OF JAVASCRIPT

Definition

- Regular expressions, often shortened to "regex" or "regexp", are patterns that help programmers match, search, and replace text. Regular expressions are powerful, but can be difficult to understand because they use so many special characters.

Exact match

To match an exact text, type in the text on the forward slash

Example if you want to match the text “help”.

Const helpReg = /help/

Flags in Regex

Regular expressions can take flags to modify their behavior. For instance, the i flag can be used to make the expression ignore case, causing it to match hello, HELLO, and Hello for the expression /hello/.

Const hello = /hello/i

The match method

- Strings have a .match() method, which accepts a regular expression as an argument and determines if the string matches that expression. It returns an array of the match strings.
- Const callRegex = /help/
- Const alarm = ‘please get me a help’
- alarm.match(callRegex)

The test method

- Instead of using the .match() method, you can use the .test() method of a regular expression to test if a string matches the pattern. Unlike .match(), .test() returns a boolean value indicating whether or not the string matches the pattern.
- Const myName = ‘Francis’
- Const nReg = /franc/
- nReg.test(myName)

Alternative sequence

- The alternate sequence | can be used to match either the text on the left or the text on the right of the |. For example, the regular expression /yes|no/ will match either yes or no.
- const birds = ‘I like turkey’
- const bReg = /chicken| turkey/
- Birds.match(bReg)

Character class

- A character class is defined by square brackets, and matches any character within the brackets. For example, [aeiou] matches any character in the list aeiou. You can also define a range of characters to match using a hyphen. For example, [a-z] matches any character from a to z.
- Const vowels = /[aeiou]/
- Const alphabets = /[a-z]/
- const stdRegex = /[0-9] students/i;

Match More than one

- To match this, the + quantifier can be used - this matches one or more consecutive occurrence. For example, the regular expression /a+/ matches one or more consecutive a characters.
- Const numReg = /[0-9]+/

Capture group

- A capture group is a way to define a part of the expression that should be captured and saved for later reference. You can define a capture group by wrapping a part of your expression in parentheses. For example, /h(i|ey) students/ would match either hi students or hey students, and would capture i or ey in a group.

Optional character

- The ? quantifier matches zero or one occurrence of the preceding character or group. For example, the regular expression /colou?r/ matches both color and colour, because the u is optional.

More on optional capture

- One last thing with this expression. You don't actually need the match value from your capture group, so you can turn it into a non-capturing group. This will allow you to group the characters together without preserving the result.

- To create a non-capturing group in a regular expression, you can add ?: after the opening parenthesis of a group. For instance, (?:a|b) will match either a or b, but it will not capture the result.

Check for spaces

- To do this, start by checking for spaces before and after your pattern. You can do this by using the meta character \s, which will match spaces, tabs, and line breaks.

Beginning and end of a string match

The ^ anchor is used to match at the beginning of the string, you can use the $ anchor to match the end of the string.

```
Const userRegex =
/[a-z](?:[0-9]$/i
```