Regx - Regular expressions in JS

whole word - word

```
let waldoIsHiding = "Somewhere Waldo is hiding in this text.";
let waldoRegex = /Waldo/; // our code
let result = waldoRegex.test(waldoIsHiding);
```

or operator |

```
let petRegex = /dog|cat|bird|fish/;
```

Ignore Case While Matching - i

- the igonore case flag "i"

```
/ignorecase/i
```

Extract Matches - word

```
let extractStr = "Extract the word 'coding' from this string.";
let codingRegex = /coding/; // our code
let result = extractStr.match(codingRegex); // our code
console.log(result)

/*[ 'coding',
  index: 18,
  input: 'Extract the word \'coding\' from this string.',
  groups: undefined ] */
```

Find More Than the First Match - g

```
let twinkleStar = "Twinkle, twinkle, little star";
let starRegex = /twinkle/ig; // our code
let result = twinkleStar.match(starRegex); // our code

console.log(result)

// [ 'Twinkle', 'twinkle' ]
```

Wildcard Period .

- match hug, huh, hut, and hum, you can use the regex /hu./

Character classes [abcd]

like wilcard persiod but you limit the choises

```
let bgRegex = /b[aiu]g/;
bag big bug
let quoteSample = "Beware of bugs in the above code; I have tfhhj only proved it correct, not tried it.";
let vowelRegex = /[aeiou]/gi; // our code
let result = quoteSample.match(vowelRegex); // our code
console.log(result);
// [ 'e',
//   index: 1,
//   input: 'Beware of bugs in the above code; I have tfhhj only proved it correct, not tried it.',
//   groups: undefined ]
```

EX - matching rang - 1-15

For example, to match lowercase letters a through e you would use `[a-e]`

`/[0-5]/` matches any number between 0 and 5, including the 0 and 5.

```
let quoteSample = "Blueberry 3.141592653s are delicious.";
let myRegex = /[h-s2-6]/gi; // our code
let result = quoteSample.match(myRegex); // our code
console.log(result)
/*
[ 'l','r','r','3','4','5','2','6','5','3','s','r','l','i','i','o','s' ]
*/
```

a set of characters that you do not want to match - ^

- To create a negated character set, you place a caret character (^) after the opening bracket and before the characters you do not want to match.
- regex that matches all characters that are not a number or a vowel.
- In an earlier challenge, you used the caret character (^) inside a character set to create a negated character set in the form [^thingsThatWillNotBeMatched]. Outside of a character set, the caret is used to search for patterns at the beginning of strings.

```
let quoteSample = "3 blind mice.";
```

match char one or more time in a row - +

```
let difficultSpelling = "Mississippi";
let myRegex = /s+/g; // our code
let result = difficultSpelling.match(myRegex);
```

Match Characters that Occur Zero or More Times - *

```
// Only change code below this line
let chewieRegex = /Aa*/g; // our code
// Only change code above this line

let result = chewieQuote.match(chewieRegex);
```

Find Characters with Lazy Matching ?

- Regular expressions are by default greedy, so the match would return longest/largest match upto a char we use ?

```
let text = "<h1>Winter is coming</h1>";
let myRegex = /<.*?>/ig; // our code
let result = text.match(myRegex);
console.log(result)
```

Match All Letters and Numbers and Negate

- The closest character class in JavaScript to match the alphabet is `\w`. This shortcut is equal to `[A-Za-z0-9_]`. This character class matches upper and lowercase letters plus numbers. Note, this character class also includes the underscore character (_).
- You can search for the opposite of the `\w` with `\W`. Note, the opposite pattern uses a capital letter. This shortcut is the same as `[^A-Za-z0-9_]`.

```
let longHand = /[A-Za-z0-9_]+/;
let shortHand = /\w+/;
```

Example- Usernames

- Criteria for user names
  - alphnumeric
  - end with number `$`, 0 or more, cannot start with number
  - lowercse and uppercase character
  - at least two apha cahr long at start `^`

```
let myUserRegx = /^[a-z][a-z]+\d*$|^[a-z]\d\d+$/i
```

`\s & /S` - Match space

- `\s` matched white space tab new line character `[\r\t\f\n\v]`
- `\S` nagation of \s `[^\r\t\f\n\v]`

```
let sample = `Whitespace is
important in separating   words`;
let countWhiteSpace = /\s/g;
let countNonWhiteSpace = /\S/g;
//countWhiteSpace => [ ' ', ' ', '\n', ' ', ' ', ' ', ' ', ' ' ]
/* countWhiteSpace => [ 'W','h','i','t','e','s','p','a','c','e','i',
's','i','m','p','o','r','t','a','n','t','i','n','s','e','p','a','r',
'a','t','i','n','g','w','o','r','d','s' ]*/
```

Match range {}

- {lower, uppper} range
- {exact}
- {only lower rage, }

```
let A4 = "aaaah";
let A2 = "aah";
let multipleA = /a{3,5}h/;
multipleA.test(A4);
multipleA.test(A2);
// True, False
```

? zero or one

- British Color
- America Colour
- to accept both `u` can be optionl i.e none or one
- `/colou?r/`

LookAhead

- Positive LookAhead `(?=[a-z])`
  - will look ahead to see if the pattern exists
- Negative LookAhead `(?!=[a-z])`
  - look ahead for patter missMatch

LookAhead

- Positive LookAhead `(?=[a-z])`
  - will look ahead to see if the pattern exists
- Negative LookAhead `(?!=[a-z])`
  - look ahead for patter missMatch

Capture group -> find repeated sub string

- indicate the group in () `(\d*)`
- user the capture group when needed as \1 \2 and so on based on capture group number.

Example - search and replace

- js has a replace method
- take string replaces with other on a string obj

```
let a = /Colour/g
let b = "a Colour b android Colour c Colour"
let c  = b.replace(a,'Qasim')
//a Colour b android Colour c Colour
//a Qasim b android Qasim c Qasim
let hello = "    Hello, World!  ";
let wsRegex = /(\s) \1*/g; // Change this line
let result = hello.replace(wsRegex,''); // Change this line
//Hello, World!
```