## Mencari kata

let myString = "Hello, World!"; // tumpukan jerami

let myRegex = /Hello/; // Jarum

let result = myRegex.test(myString); // mencari jarum di tumpukan jerami

## Literal Match (sama persis)

-sama dengan mencari kata

## OR Operator ( | )

let petString = "James has a pet cat.";

let petRegex = /cat**|**dog**|**bird**|**fish/; // Change this line

let result = petRegex.test(petString);

## Ignore case (huruf besar/kecil) (i)

let myString = "freeCodeCamp";

let fccRegex = /freecodecamp/**i**; // Change this line

let result = fccRegex.test(myString);

## Mengambil apa yang match nya (.match)

Berbentuk **array**, jika tidak ada berbentuk **null.**Syntax .match() **kebalikan** dengan .test()

'string'.match(/regex/);

/regex/.test('string');

// ==========================================================

let extractStr = "Extract the word 'coding' from this string.";

let codingRegex = /coding/; // Change this line

let result = extractStr.match(codingRegex); // Change this line

## Cari lebih dari kata match pertama (Berapa kali dia ketemu). (g)

Cari berapa kali kata tersebut muncul. Tambahkan (i) untuk mengabaikan besar kecil huruf.

let twinkleStar = "Twinkle, twinkle, little star";

let starRegex = /twinkle/gi; // Change this line

let result = twinkleStar.match(starRegex); // Change this line

## Mencari apappun setelah titik (.)

Yang terpenting kita beri huruf di depan/belakangnya apa. Contoh mencari hug, huh, hut, hum cukup gunakan regex /**hu./**

**Mencari run,sun,fun,pun,nun & bun**

let exampleStr = "Let's have fun with regular expressions!";

let unRegex = /.un/; // Change this line

let result = unRegex.test(exampleStr);

## Mencari satu karakter dengan berbagai kemungkinan.

Menambahkan *character classes*, grup karakter dengan membungkusnya dengan [ ]. Contoh mencari bag, big, bug tapi tidak mau bog maka **/b[aiu]g/**.

**Mencari aiueo**

let quoteSample = "Beware of bugs in the above code; I have only proved it correct, not tried it.";

let vowelRegex = /[a,i,u,e,o]/ig; // Change this line

let result = quoteSample.match(vowelRegex); // Change this line

## Mencocokan Huruf yang ada di Alphabet

**[a-z]** bisa juga mencari dengan range misal **a** sampai **e [a-e]**.

let quoteSample = "The quick brown fox jumps over the lazy dog.";

let alphabetRegex = /[a-z]/gi; // Change this line

let result = quoteSample.match(alphabetRegex); // Change this line

## Menggabungkan Number dan Letter Alphabet

Menggunakan hyphen **(-)** untuk untuk range juga bisa dugunakan untuk number. [0-5] Jika alphabet dan number jadi **[a-z0-9]** ditulid dalam satu kurung siku.

**Huruf antar h-s dan 2-6**

let quoteSample = "Blueberry 3.141592653s are delicious.";

let myRegex = /[h-s2-6]/ig; // Change this line

let result = quoteSample.match(myRegex); // Change this line

## Mencocokan satu karakter yang terkecuali / tidak mau dicocokan.

*Negated characer* menggunakan carret **(^)** setelah kurung siku buka & ttutup **/[^aiueo]/** mencocokan yang bukan huruf vokal / aiueo.

**Mencari karakter yang bukan number dan huruf vokal.**

let quoteSample = "3 blind mice.";

let myRegex = /[^0-9aiueo]/gi; // Change this line

let result = quoteSample.match(myRegex); // Change this line

## Mencocokan karakter yang muncul satu atau lebih (berulang)

Menggunakan plus **(+).** Contoh kita menuliskan **/a+/g** jika saling berdekatan contoh **aaabc** akan mengembalikan **[“aaa”].** Tetapi jika terpisah contoh **abab** akan mengembalikan **[“a”, “a”]  
aabc = [“aa”]  
abab = [“a”, “a”]**

**Mencocokan huruf s dalam kata Mississippi**

let difficultSpelling = "Mississippi";

let myRegex = /s+/gi; // Change this line

let result = difficultSpelling.match(myRegex);

## Mencocokan karakter yang muncul nol atau lebih

Menggunakan asterisk atau bintang **(\*).** Contoh kita menuliskan **/go\*/** lalu ada kata “gooooooool!”, “gut”, “over the moon”. Akan mengembalikan [“goooooooo”], [“g”] & null.

// Only change code below this line

let chewieQuote = "Aaaaaaaaaaaaaaaarrrgh!"

let chewieRegex = /Aa\*/; // Change this line

// Only change code above this line

let result = chewieQuote.match(chewieRegex);

## Mencari karakter menggunakan *Lazy Matching*

*Greedy match =* Kemungkinan terpanjang & *Lazy Match =* Kemungkinan terpendek. Jika ketemu pertama dia tidak akan melanjutkan

Contoh *Greedy match*  kita menuliskan **/t[a-z]\*i/** dalam string “titanic”. Regex ini polanya diawali denan huruf **t** diakhiri dengan **i** dan beberapa huruf di tengahnya. Ini akan mengembalikan **[“titani”]**

Contoh *Lazy match* dengan menggunakan tanda tanya **?** dapat mengubanya menjadi lazy match string “titanic” dengan di regex **/t[a-z]\*?i/** mengembalikan **[“ti”].**

Jika string nya “t2itanic” maka akan mengembalikan “tanic”, karena dalam regex nya huruf depan **t** ditengahnya **alphabet** diakhiri **i**, sedangkan “t2i” ditengahnya ada number jadi tidak dianggap.

**Hanya mengembalikan <h1> saja tidak dengan isi textnya** karena ini lazy </h1> tidak di anggap

let text = "<h1>Winter is coming</h1>";

let myRegex = /<.\*?>/; // Change this line

let result = text.match(myRegex);

## Mencocokan pola di awal strings

Caret **(^)** jika disimpan di awal di dalam kurung siku **[]** akan **mengecualikan** karakter di dalamnya.

Tetapi jika diluar kurung siku akan mencocokan pola di awal string. (**harus diawal)**

let firstString = "Ricky is first and can be found.";

let firstRegex = /^Ricky/;

firstRegex.test(firstString);

let notFirst = "You can't find Ricky now.";

firstRegex.test(notFirst);

diatas akan mengembalikan **true** untuk yang atas karena Ricky ada di awal dan **false** untuk yang bawah karena Ricky ada di tengah

**Caret regex untukk mencari Cal di awal string**

 let rickyAndCal = "Cal and Ricky both like racing.";

let calRegex = /^Cal/; // Change this line

let result = calRegex.test(rickyAndCal);

## Mencocokan pola di akhir strings

Menggunakan *dollar sign* $ di akhir regex

let theEnding = "This is a never ending story";

let storyRegex = /story$/;

storyRegex.test(theEnding);

let noEnding = "Sometimes a story will have to end";

storyRegex.test(noEnding);

diatas akan mengembalikan **true** untuk yang atas karena story di akhir dan **false** untuk yang bawah karena story ditengah.

**Gunakan $ untuk mencocokan caboose di akhir string**

let caboose = "The last car on a train is the caboose";

let lastRegex = /caboose$/; // Change this line

let result = lastRegex.test(caboose);

## Mencocokan semua huruf dan angka

Untuk mencocokan alphabet adalah **\w.** shortcut itu sama dengan **[A-Za-z0-9\_].** Class karakter ini mencocokan huruf besar dan huruf kecil juga angka. Perlu diingat character class ini juga memasukan underscore **( \_ ).**

let longHand = /[A-Za-z0-9\_]+/;

let shortHand = /\w+/;

let numbers = "42";

let varNames = "important\_var";

longHand.test(numbers);

shortHand.test(numbers);

longHand.test(varNames);

shortHand.test(varNames);

keempat test diatas akan mengembalikan **true.**

## Mencocokan semua kecuali huruf dan angka

Kebalikan dari \w adalah **\W** (kapital) sama dengan **[^A-Za-z0-9\_].**

let shortHand = /\W/;

let numbers = "42%";

let sentence = "Coding!";

numbers.match(shortHand);

sentence.match(shortHand);

match pertama akan mengembalikan **[“%”]**, dan kedua akan mengembalikan **[“!”]**.

## Mencocokan hanya angka saja

Menggunakan **\d** dengan d kecil sama dengan **[0-9]**

**gunakan \d untuk menghitung berapa digit judul film, yang berisi number.**

let movieName = "2001: A Space Odyssey";

let numRegex = /\d/g; // Change this line

let result = movieName.match(numRegex).length;

## Mencocokan kecuali angka

Kebalikan dari \d dengan menggunakan **\D** (kapital) sama dengan [^0-9]

let movieName = "2001: A Space Odyssey";

let noNumRegex = /\D/g; // Change this line

let result = movieName.match(noNumRegex).length;

## Mencocokan Spasi

Menggunakan **\s** s kecil sama dengan **[ \r\t\f\n\v]**

let whiteSpace = "Whitespace. Whitespace everywhere!"

let spaceRegex = /\s/g;

whiteSpace.match(spaceRegex);

match diatas akan mengembalikan [“ ”,” ”]

## Mencocokan yang bukan Spasi

Kebalikan dari \s dengan menggunakan **\S** (kapital) sama dengan **[ ^\r\t\f\n\v]**

let whiteSpace = "Whitespace. Whitespace everywhere!"

let nonSpaceRegex = /\S/g;

whiteSpace.match(nonSpaceRegex).length;

match diatas akan mengembalikan length sebesar 32

## Mencocokan range atas dan bawah Nomor

Menggunakan curly brackets **{ }** dan memasukan 2 nomor di dalamnya untuk range angka awal dan angka akhir.

Contoh mencocokan huruf a yang muncul 3 sampai 5 kali dalam string ah, regexnya yaitu **/a{3,5}h/**

let A4 = "aaaah";

let A2 = "aah";

let multipleA = /a{3,5}h/;

multipleA.test(A4);

multipleA.test(A2);

test awal akan mengembalikan **true** dan test kedua akan mengembalikan **false**

**mencocokan kata “Oh no” ketika h nya ada 3 sampai 6 kali.**

let ohStr = "Ohhh no";

let ohRegex = /Oh{3,6} no/g; // Change this line

let result = ohRegex.test(ohStr);

## Mencocokan hanya nomor terkecil(lower) saja

Untuk mencocokan nomor terkecil saja, tuliskan **nomor pertama** diikut dengan tanda coma **( , ).**

Contohnya dalam string “hah” huruf a muncul setidaknya 3 kali / 3 sampai unlimited, regexnya adalah **/ha{3,}h/**

let A4 = "haaaah";

let A2 = "haah";

let A100 = "h" + "a".repeat(100) + "h";

let multipleA = /ha{3,}h/;

multipleA.test(A4);

multipleA.test(A2);

multipleA.test(A100);

menurut urutan diatas akan mengembalikan true, false, true.

**Mencocokan string “Hazzah” hanya ketika z nya minimal ada 4.**

let haStr = "Hazzzzah";

let haRegex = /Haz{4,}ah/; // Change this line

let result = haRegex.test(haStr);

## Mencocokan nomor hanya x kali saja

Dengan hanya memasukan satu nomor didalam *curly bracket* **{ }.**

Contohnya untuk mencocokan string hah dengan huruf a 3 kali saja regexnya **/ha{3}h/**

let A4 = "haaaah";

let A3 = "haaah";

let A100 = "h" + "a".repeat(100) + "h";

let multipleHA = /ha{3}h/;

multipleHA.test(A4);

multipleHA.test(A3);

multipleHA.test(A100);

mengikuti urutan test diatas akan mengembalikan false, true, false.

**Mencocokan string “Timber” hanya ketika m nya ada 4.**

let timStr = "Timmmmber";

let timRegex = /Tim{4}ber/; // Change this line

let result = timRegex.test(timStr);

## Check semua atau tidak sama sekali

Kadang pola yang akan dicari kemungkinan ada atau tidak ada sama sekali, untuk mencari kemungkinan kemunculan dari sebuah elemennya menggunakan tanda tanya ( **?** ). Fungsinya untuk mencari 0 atau satu dari kemunculan element selanjutnya. Kita bisa berpikir simbol ini sebagai element sebelumnya optional.

Contohnya ada perbedaan penulisan antara Amerika dan Inggris, dan kita bisa menggunakan tanda tanya untuk mencocokan kedua ejaan.

let american = "color";

let british = "colour";

let rainbowRegex= /colou?r/;

rainbowRegex.test(american);

rainbowRegex.test(british);

tanda tanya di depan u artinya u boleh ada boleh tidak. Kedua test diatas akan mengembalikan true.

## Pandangan ke Depan Positif dan Negatif

*Lookahaeads* adalah sebuah pola yang memberitahun JavaScript untuk melihat kedepan dalam string, untuk mencheck polanya kedepan. Ini bisa berguna ketika akan mencari beberapa pola dalam string yang sama. Ada dua jenis lookaheads: Positif lookahead dan negatif lookahead.

Penulisannya menggunakan **(?=...)** dimana ( **... )** dibutuhkan tidak cocok asdasd

Part yang ada di dalam lookahead tidak akan diambil (tapi dibutuhkan)

let quit = "qu";

let noquit = "qt";

let quRegex= /q(?=u)/;

let qRegex = /q(?!u)/;

quit.match(quRegex);

noquit.match(qRegex);

quRegex = jika ada u, u nya jangan diambil qRegex = jika bukan u jangan diambil, keduanya match diatas akan mengembalikan [“q”].

**Password Checker mencocokan panjang karakternya lebih dari 5 karakter & harus mempunyai 2 nomor yang berurutan**

let sampleWord = "astronaut22";

let pwRegex = /(?=\w{6,})(?=\w\*\d{2})/; // Change this line

let result = pwRegex.test(sampleWord);

**(?=\w{6,})** mengecek panjang karakter lebih dari 5  
(?=\w\*\d{2}) mengecek karakter harus memiliki 2 nomor yang berurutan dan minimal 0 bilangan / huruf sebelum angka  
() tanda kurung menandakan grouping  
result diatas akan mengembalikan true.

Intinya lookahead mencari **dimanapun** ke depan yang berhasil. Dalam contoh diatas jika 22 ada ditengah return nya true.

Jika kita hapus tanda ?= lalu menyimpan 22 nya di tengah return nya jadi false.

## Mencheck Grouping campuran dari karakter

Terkadang kita ingin memeriksa grup karakter menggunakan regular expression dan caranya adalah menggunakan parentheses / tanda kurung **( ).**

Jika ingin mencari Penguin atau Pumpkin dalam sebuah string, bisa menggunakan Regular expression: **/P(engu|umpk)in/g,** lalu check dengan menggunakan test().

let testStr = "Pumpkin";

let testRegex = /P(engu|umpk)in/;

testRegex.test(testStr);

test() diatas akan mengembalikan true

**Periksa sebuah regex dengan nama Franklin Roosevelt atau Eleanor Roosevelt dalam aturan case sensitive dan dibolehkan menambahkan nama tengah.**

let myString = "Eleanor D. Roosevelt";

let myRegex = /(Franklin|Eleanor).\*Roosevelt/; // Change this line

let result = myRegex.test(myString); // Change this line

// After passing the challenge experiment with myString and see how the grouping works

**.\*** akan memeriksa 0 sampai 1 (tidak ada juga tidak apa-apa) kata apapun setelah nama depan dan sebelum rossevelt.

## Menggunakan Capture Groups untuk mencari dan mengganti

Dengan menggunakan **.replace()** pada string. Lalu berikan 2 parameter, parameter pertama adalah regex nya parameter kedua adalah ingin diganti dengan apa jika ketemu polanya.

let wrongText = "The sky is silver.";

let silverRegex = /silver/;

wrongText.replace(silverRegex, "blue");

replace() diatas akan mengembalikan “The sky is blue”

menukar string

"Code Camp".replace(/(\w+)\s(\w+)/, '$2 $1');

Replace() diatas akan mengembalikan (Camp Code)  
(\w+) mengambil kata pertama yaitu Code, \s spasi, (\w+) mengambil kata kedua yaitu Camp  
$2 $1 menukar