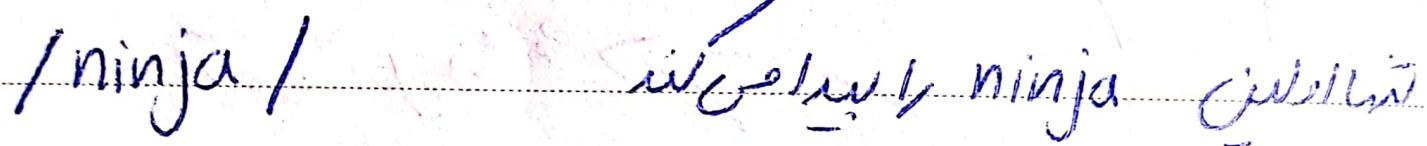


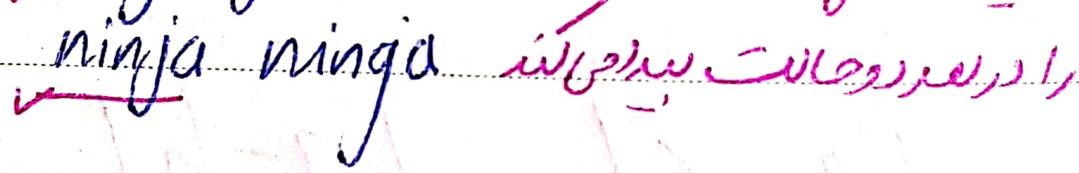
regular expression (online)

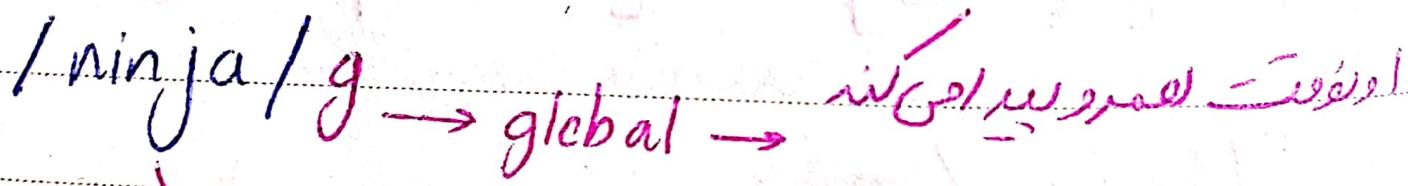
regex101.com

1 Pattern 1

/ninja/ 

ninjDojo 

ninja ninja 

/ninja/g → global → 

case sensitive

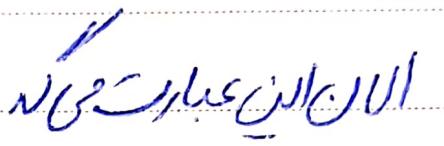
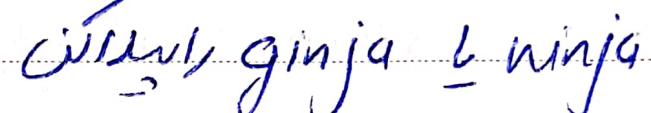
/ninja/i → insensitive case

/ninja/gi → global

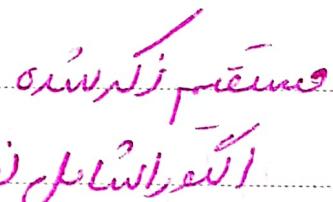
/ninja/gi → insensitive

character set [ ] brackets

ninja / ginja

/[ng]inja/   
ninja -> 

exclude set [^ ...]

/[^pe]ooo/   
oiii →   
2000 e000 p000  
✓ X X

Range [a-z] [a-h] [g-n]

/[c-i]inja/g

Date:

Range Case in Sensitive

[a-zA-Z]

/ [a-zA-Z]inja /

Range number [0-9]

/ [0-9]inja / gi

[0-9] +

/ [0-9] + /

consecutive numbers +  
single digit  
grouped consecutive numbers

0 01 981 888 1111 ---

1 09

9 91

99

case sensitive  
means [a-9] is different  
from 9

No.

Date:

برای متنی که از مجموعه زیر میگذرد

1 [0-9] {11} /gi

برای متنی که از مجموعه زیر میگذرد

1 [a-z] {5,8}

برای متنی که از مجموعه زیر میگذرد

1 [a-z] {5, }

No. \_\_\_\_\_  
Date: \_\_\_\_\_

## Meta characters in Regex

- \d match any digit character (Same as [0-9])

- \w match any word character (a-z, A-Z, 0-9 and \_)

- \s match a whitespace character (spaces, tabs etc)

- \t match a tab character only

d -- matches the literal character, 'd'

\d -- matches any digit character

/ \d \s \w /gi

I  $\longleftrightarrow$  Z

/ \d \f{m} \s \w /gi

in this code \f{m} is new pattern

## Special character

'+' The one-or-more quantifier

'\'' The escape character

'[ ]' The character set

'[ ^ ]' The negate symbol in a character set

No.  
Date:

## Escape characters

To insert characters that are illegal in a string, you must use an escape character.

An escape character is a backslash \ followed by the character you want to insert.

An example of an illegal character is a double quote inside a string that is surrounded by double quotes:

```
str ← "We are the so-called "Vikings",  
      from the north."
```

str.

## Special character

171 The zero-or-one quantifier  
(makes a preceding char optional)

!! Any character whatever (except the newline character)

\* The 0-or-more quantifier (a bit like +)

/hello?/      hell ✓  
      └─────────┐  
          is  
      optional      hello ✓

/a[a-Z]?/      a → match !  
                  n  
                  optional      az → match !

|Car.|      X Car → not match  
↓ any char      car@ → ✓ match  
except newline

$/[a-z]^*/g$

$\underbrace{/[a-z]^*/g}_{\text{a}}$  →  $a \rightarrow \text{match}$

abadan → match

abb → match

$/abc^*/$  →  $abc^* \rightarrow \text{match}$

$/abc^.$  →  $abc. \rightarrow \text{match}$

$/abc^+$  →  $abc^+ \rightarrow \text{match}$

1 Start form field

\$ end form field

$/^{[a-z]}\{5\} \$/$  میتوانی  
نمیتوانی

Start

end

$/[a-z]\{5\} \$/$

195 821Xchni → match

# Alternate Characters.

P|t

Port

/P|t/

P → match  
t → match

/P|tyre/

P → match

Pyre → not match

tyre → match

| → Pipe character

/Pyre|tyre/

|(P|t)yre/

Pyre → match  
tyre → match

↓  
jilp  
kōjūw  
(s)p  
tōtōs

( ) ? ئىچىرىنىڭ ئەللىكىنىڭ  
نىتى رېگىزىنەنىڭ mini اىنۇنىڭ  
expression

/ (pet | toy | crazy) rabbit

Pet — rabbit match!  
toy — rabbit match!

/ (pet | toy | crazy)? rabbit /  
optional

## Making RegEx in JavaScript

Var reg = / / ;

duzus

Var reg = /[a-z]/ig ; ↗

perus

Var reg2 = new RegExp (reg, 'ig') ;

new regular expression

# Telephone RegEX

const patterns = {

telephone: /\d{11}\$/,

telephone: /\^ \d{11}\$/,  
};

## Testing a RegEx Pattern

const inputs = document.querySelectorAll("input");

input.addEventListener just for a method

inputs.forEach((()=>{}))

```
const inputs = document.querySelectorAll('input');
```

```
const patterns = {
```

```
    telephone: /\^[\d\$]{11}\$/
```

```
}
```

```
inputs.forEach((input) => {
```

```
    input.addEventListener('keyup', (e) => {
```

```
        console.log(e.target.attributes.name.value);
```

```
    });
```

```
// Validation function
```

```
function validate(field, regex) {
```

```
    regex.test(field.value)
```

```
}
```

// Validation script here

```
const inputs = document.querySelectorAll('input');
```

Const pattern = f

```
telephone: /\d{11}$/ ;
```

// Validation function

```
function validate (field, regex){
```

```
    console.log(regex.test(field.value));
```

```
}
```

inputs.forEach((input) => {

```
    input.addEventListener('keyup', (e) => {
```

```
        //console.log(e.target.attributes.name.value);
```

```
        validate(e.target,
```

```
            patterns[e.target.attributes.name.value]
```

```
    });
```

```
});
```

```
function validate(field, regex){  
    if(regex.test(field.value)){  
        field.className = 'valid';  
    } else {  
        field.className = 'invalid';  
    }  
}
```

\*\*\*

```
const patterns = {
```

```
telephone: /\d{11}$/,
```

```
username: /[a-zA-Z]\w{5,12}$/,
```

```
password: /\w{6-}$/,
```

```
};
```

Profile slug: http://my-profile

the portion of a review profile URL that is unique to your business

No.  
Date:

# Email RegEx Pattern

email : /^([a-zA-Z\d]+\.[a-zA-Z\d]+@[a-zA-Z\d]+\.{2,8})+(\.\[a-zA-Z\]{2,8})?\$/