regular expression (online)
regex101.com

/ Pattern /

/ninja/ تنها اولین ninja را پیدا می‌کند

ninjDojo
ninja ninja

اولین ninja را در هر دو حالت پیدا می‌کند

/ninja/g → global → در هر دو حالت پیدا می‌کند
↳ case sensitive

/ninja/i → in sensetive case

/ninja/gi → global → in sensitive

No.
Date:

character set [ ] brackets

ninja / ginja

/[ng]inja/ الان این عبارت می‌گه ginja یا ninja رو بیدا کن

exclude set [^ ...]

/[^pe]ooo/ مستقیم ذکر شده → اگه در اشتباه نفی میشود

2ooo ✓ eooo ✗ pooo ✗

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

/[c-i]inja/g

Date:

Rage Case in Sensitive

[a-zA-Z]

/[a-zA-Z]inja/

Range number [0-9]

/[0-9]inja/gi

[0-9]+

/[0-9]+/

+ بالای حی گوید تعدد ارقام عددی حداقل از یک بار هیچ است.

حداقل ارقام عددی با یک عدد کاراکتر عددی یا بالاتر یعنی

0 01 ۹۲۱ ۲۹۴ ۸۷۱۴ ----

1 09

2

9 91

99

تعدادی کاراکترهای عددی حداقل یک بار از [0-9] استفاده می شود.

No.
Date:

برای شماره تلفن با ۱۱ عدد میتوان از عبارت زیر استفاده کرد.

/[0-9]{11}/gi

برای متونی بین ۵ تا ۸ کاراکتر می توان از دستور زیر استفاده کرد

/[a-z]{5,8}

برای متونی با حداقل ۵ کاراکتر به بالا از دستور زیر استفاده کرد

/[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)
- \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

Date:

/\d\s\w/gi

1 ⟷ Z

/\d{3}\s\w/gi

می‌خواهم سه حرف اول من عدد باشد.

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 surronded by double quotes:

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

No.
Date:

special character

'?' 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?/ → o is optional

hell ✓
hello ✓

/a[a-Z]?/ → optional

a → match !
az → match !

/car./ → any char except newline

✗ car → not match
car@ → ✓ match

/[a-z]*/g

/a[a-z]*/g → a → match
abadan → match
abb → match

/abc*/ abc* → match
/abc\./ abc. → match
/abc\+/ abc+ → match

^ start form field
$ end form field

/^[a-z]{5}$/ دقیقاً ۵ حرف باشد
نه کمتر و نه بیشتر

start
end

/[a-z]{5}$/

آخر عبارت
195821xcnni → 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

↓ پرانتز استفاده کنید برای گروه‌بندی

No.
Date:

(   ) آخر مابین پرانتز می آید در واقع چیست؟
یک mini Regex مستقلی است
expression.

/(pet|toy|crazy) rabbit

Pet _ rabbit match!
toy _ rabbit match!

/(pet|toy|crazy)? rabbit/

optional

Making RegEx in JavaScript

```
var reg = //;
Var reg = /[a-z]/ig;
```
روش اول

روش دوم
```
Var reg2 = new RegExp(reg, 'ig');
```
یعنی هر دوتا را با هم می گیرد یا خیر

No.
Date:

Telephone RegEX

```
const patterns = {
  telephone: /^\d{11}?$/
  telephone: /^\d{11}?$/
};
```

Testing a RegEx Pattern

```
Const inputs = document.querySelectorAll("inputs");
input.addEventnn> just for a method
inputs.forEach(() => {})
```

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

Const patterns = {
      telephone: /^\d{11}$/
};

inputs.forEach((input) => {

    input.addEventListener ('keyup',
                    (e) => {
    Consele.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 = {
    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}$/,
    username: /^[a-z\d]{5,12}$/,
    password: /^[\w@-]$/
};
```

Profile slug: http url /my-profile
the portion of a review profile URL that is unique to your business

No.
Date:

Email RegEx Pattern

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