

## Рядки. Регулярні вирази

<https://uk.javascript.info/string>

[https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global\\_Objects/String](https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/String)

[https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global\\_Objects/RegExp](https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/RegExp)

<https://itwiki.dev/front-end/js-reference/javascript/jsref-obj-regexp-asp>

<https://regex101.com/>

<https://regexper.com/>

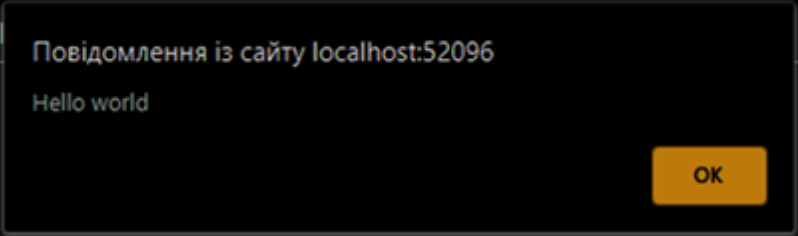
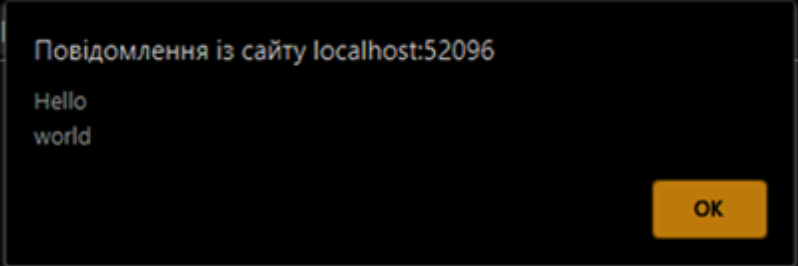
## Створення рядків

Створення з використанням літералу	Приклад	Визначення типу
let <code>змінна</code> = '... текст ...'	let s='Hello' let s2 = "Hello"	<code>typeof(s)</code> 'string'

Створення з використанням конструктора	Приклад	Визначення типу
let <code>змінна</code> = new String("... текст ...")	let s=new String('Hello')	<code>typeof(s)</code> 'object'

(бажано не використовувати рядки як об'єкти, оскільки це призводить до сповільнення роботи програми)

## Літерали рядків

	Загальна форма	Приклад
Запис літералів за допомогою одинарних чи подвійних лапок	<code>`рядок_тексту`</code> <code>"рядок_тексту"</code>  <code>var змінна = "рядок_тексту"</code>	Літерал у одинарних лапках: <code>'Це рядок'</code> Літерал у подвійних лапках: <code>"це теж рядок"</code> <b>Відкриваючі і закриваючі лапки повинні бути ідентичні:</b> <code>'помилка'</code> , <code>"теж помилка"</code>  Змінна рядкового типу: <code>var str='Приклад';</code>
Запис великих літералів у декількох рядках: <ul style="list-style-type: none"><li>записати в кінці символ <code>«\»</code>;</li><li>використовувати шаблони текстів (записувати рядок у символах <code>" ` "</code>); при цьому зберігаються також символи переходу на новий рядок.</li></ul>	<code>"перша частина тексту \</code> <code>друга частина тексту"</code>  <code>'перша частина тексту \</code> <code>друга частина тексту'</code>	<code>var s = "Hello \</code> <code>world";</code> <code>alert(s);</code>   ----- <code>var s = `Hello</code> <code>world`;</code> <code>alert(s);</code>  

Ескейп послідовності	прості ескейп-послідовності – службові символи починаються з символу «\»	\n - перехід на новий рядок, \t - горизонтальна табуляція, \r – повернення каретки			
Символи (незалежно від таблиці кодування, яка була використана при написанні скрипта, рядки завжди кодуються з використанням Unicode ). Символи можуть бути вставлені у текст за допомогою їх кодів у таблиці кодування Unicode у форматі \uNNNN	ескейп-послідовності Unicode – символи «\u», за якими вказують код символу з чотирьох цифр у шістнадцятковій системі числення	Наприклад, \u00A9 - символу копірайту ©  Приклад: <table><tr><td>U+270C</td><td>👉</td><td>Victory hand</td></tr></table>  alert('\u270C')	U+270C	👉	Victory hand
U+270C	👉	Victory hand			
Екранування символів Якщо необхідно вставити у рядок деякий символ, який має спеціальне призначення, то необхідно його «екранувати» за допомогою символу “\”	<div>\</div> <div>символ_спеціального_призначення</div>	\' – апостроф \" – подвійні лапки  var s1='здоров\'я', s2="готель \"Ужгород\" ", s3="здоров'я";			

### Створення рядків

Створення рядка, як багаторазової копії заданого фрагмента	<code>рядок . repeat ( кількість_повторень )</code>	<pre>let hello = "hello "; console.log(hello.repeat(3)); //hello hello hello</pre>
--	---	--

### Заповнення рядків

Розширює рядки на початку і заповнює додаткові позиції вказаним заповнювачем (або пробілом, якщо не вказано)	<code>padStart ( кінцева_довжина )</code> <code>padStart ( кінцева_довжина, заповнювач )</code>	<pre>let s='Hello'; let s7_space=s.padStart(7) //'  Hello'  let s7_0=s.padStart(7,0)  // '00Hello'</pre>
Розширює рядки в кінці і заповнює додаткові позиції вказаним заповнювачем (або пробілом, якщо не вказано)	<code>padEnd ( кінцева_довжина )</code> <code>padEnd ( кінцева_довжина, заповнювач )</code>	<pre>let s='Hello'; let s2_space= s.padEnd(7)  // 'Hello  ' let s2_0= s.padEnd(7, 0)   // 'Hello00'</pre>



Задача. Користувач вводить ім'я. Створити рядок з 15 повторень цього рядка.

Задача. Вводиться кількість годин і хвилин. Вивести у форматі "години:хвилини" при цьому додати 0 якщо величини менше 10.

Задача. Вивести на екран символ 

Задача. Дано масив імен. Вирівняти усі імена за довжиною найбільшого.

<p>Шаблони рядків (шаблони дозволяють вставити значення у рядок); шаблони записуються у похилих лапках</p> <div>текст</div> <p>значення вставляють у фігурних дужках, перед якими записують символ “\$”</p> <div>... \${значення} ...</div>	<pre>`текст \${ значення_для_вставки } текст`</pre>	<pre>let name = "Tom"; let hello = `Hello \${name}`; console.log(hello);      // Hello Tom  let age = 23; let info = `\${name} is \${age} years old`; console.log(info);      // Tom is 23 years old</pre>
<p>JavaScript дозволяє передати у функцію шаблон рядка (при цьому передаються усі динамічно обчислювані фрагменти як окремі параметри). Це дозволяє при вставці значень у шаблон виконувати над ними деякі операції</p>	<pre>----- опис функції --- function функція(     список_текстових_фрагментів,     знач.1, знач2, ... ){     ...Формування кінцевого результату... }  --- виклик функції --- функція`... шаблон...`  - список_текстових_фрагментів - це масив з фіксованими частинами тексту шаблону - знач.1, знач2, ... - це значення, які вставляємо у шаблон</pre>	<pre>const person = 'Ivan' const age = 12  function check(parts, name, age) {     console.log(parts)     // ['Hello ', '! You are ', ' years old!']     console.log(name) // 'Ivan'     console.log(age) // 12     return parts[0] + name.toUpperCase() + parts[1]     + (age + 2) + parts[2] }  let checkedTemplate = check`Hello \${person}! You are \${age} years old!` console.log(checkedTemplate) // Hello IVAN! You are 14 years old!</pre>

Задача. Дано ім'я користувача, стать і вік. З використанням шаблонів вивести на екран рядок за шаблоном  
{пан/пані} {ім'я} : {пенсіонер/не пенсіонер}



### Довжина рядка

	Загальна форма	Приклад
Повертає кількість символів	<code>рядок . length</code>	<pre>let hello = "Hello world!"; let n = hello.length // n=12</pre>

### Зміна регістра

	Загальна форма	Приклад
Приведення усіх символів до нижнього регістру	<code>рядок . toLowerCase()</code>	<pre>let hello = "Hello World" console.log(hello.toLowerCase()) // "hello world"</pre>
Приведення усіх символів до верхнього регістру	<code>рядок . toUpperCase()</code>	<pre>let hello = "Hello World" console.log(hello.toUpperCase()); // "HELLO WORLD"</pre>

### Звертання до символів

	Загальна форма	Приклад
зчитування символу – <code>charAt</code>	<code>рядок . charAt( індекс )</code>	<pre>//01234567... let hello = "Hello World" console.log(hello.charAt(2)); // 1</pre>
зчитування символу з використанням індексатора (використати квадратні дужки) (рядки є незмінюваними)	<code>рядок [ індекс ]</code>	<pre>//01234567... let hello = "Hello World" console.log(hello [ 2 ] ); // 1</pre>
зчитування коду символу	<code>рядок . charCodeAt ( індекс )</code>	<pre>let hello = «Привіт»; console.log(hello.charCodeAt(2)); // 1080</pre>
одержання символу за його кодом	<code>String . fromCharCode ( код_символу )</code>	<pre>alert( String.fromCharCode(1072) ); // 'a'</pre>

Задача. Користувач вводить логін. Перевірити правильність логіна (регістр не враховується)

Задача. Шифр Цезаря.

Об’єднання

	Загальна форма	Приклад
З використанням методу <code>concat</code> (створюється новий рядок, який є об’єднанням даних)	<code>рядок1 . concat ( рядок2 )</code>	<pre>let hello = "Привет "; let world = "мир"; hello = hello.concat(world); console.log(hello); // Привет мир</pre>
З використанням оператора «+» (створюється новий рядок, який є об’єднанням даних)	<code>рядок1 + рядок2</code>	<pre>let hello = "Привет "; let world = "мир"; hello = hello + world ; console.log(hello); // Привет мир</pre>

Вибір підрядка

	Загальна форма	Приклад
Копіювання з вказаної позиції вказаної кількості символів	<code>рядок . substr(start [, length])</code>	<pre>let hello = "Привіт дім. Дому мир" let bye = hello.substr(12, 4); console.log(bye); // Дому</pre>
копіювання з початкової до кінцевої позиції (кінцева може бути більшою за початку - будуть поміняні місцями)	<code>рядок . substring(start [, end])</code>	<pre>let hello = "Привіт дім. Дому мир" let world = hello.substring(7, 10); // з 7-го до 10-го(не вкл.) console.log(world); //   дім</pre>
копіювання з початкової до кінцевої позиції (можна вказувати від’ємні індекси)	<code>рядок . slice(start [, end])</code>	<pre>let hello = "Привіт дім. Дому мир" let world = hello.slice (7, 10); // з 7-го до 10-го(не вкл.) console.log(world); //   дім</pre>

Задача. Отримати першу і другу половину введеного рядка

Задача. Отримати рядок, у якому немає першого і останнього символу

## Пошук у рядку

	Загальна форма	Приклад
визначення входження чи невходження деякого підрядка у рядку (повертає false, якщо підрядок не входить у рядок)	<code>рядок . includes ( підрядок )</code>	<pre>let hello = "привет мир. пока мир";  console.log(hello.includes("мир")); // true console.log(hello.includes("миг")); // false</pre>
пошук першого входження деякого підрядка у рядку (повертає -1, якщо підрядок не входить у рядок)	<code>рядок . indexOf ( підрядок )</code>	<pre>let hello = "привет мир. пока мир"; let key = "мир"; let lastPos = hello.indexOf(key); console.log("Первое вхождение: ", firstPos);    // 7</pre>
знаходження останнього входження деякого фрагмента (пошук з кінця)	<code>рядок . lastIndexOf ( підрядок )</code>	<pre>let hello = "привет мир. пока мир"; let key = "мир"; let lastPos = hello.lastIndexOf(key); console.log("Последнее вхождение: ", lastPos);  // 17</pre>
чи починається з вказаного фрагменту	<code>startsWith()</code>	<pre>let hello = "let me speak from my heart"; console.log(hello.startsWith("let"));           // true console.log(hello.startsWith("Let"));           // false console.log(hello.startsWith("lets"));          // false</pre>
чи закінчується вказаним фрагментом	<code>endsWith()</code>	<pre>let hello = "let me speak from my heart"; console.log(hello.endsWith("heart"));           // true console.log(hello.startsWith("bart"));          // false</pre>



Задача. Дано адресу сайту. Перевірити, чи є він з шифруванням (починається з «https»)

Задача. Дано адресу сайту. Перевірити, чи закінчується шлях з «edu».

Задача. Дано номер телефону. Перевірити, чи є цей телефон телефоном з України (починається на «+38»)

Порівняння

	Загальна форма	Приклад
Порівняння на рівність	<code>рядок1 === рядок2</code>	<code>let str1 = 'hello'</code> <code>let str2 = 'Hello'</code> <code>str1 === str2 //false</code>
Порівняння (який має бути раніше у алфавітному порядку) <, >, <=, >=	<code>рядок1 &lt; рядок2</code>  <code>рядок1 &gt; рядок2</code>	<code>let str1 = 'Allo'</code> <code>let str2 = 'Abba'</code> <code>str2 &lt; str1 //true</code>
Порівня з врахуванням національних особливостей <ul style="list-style-type: none"><li>Повертає від'ємне число, якщо <code>рядок1</code> менше, ніж <code>рядок2</code>.</li><li>Повертає додатне число, якщо <code>рядок1</code> більше, ніж <code>рядок2</code>.</li><li>Повертає 0, якщо вони рівні.</li></ul>	<code>рядок1 . localeCompare ( рядок2 )</code>	<code>let str = "Їжак"</code>  <code>alert( str.localeCompare("Ялинка") ) // -1</code>

Заміна фрагментів у рядку

	Загальна форма	Приклад
Заміна першого входження старого фрагменту тексту на новий фрагмент	<code>str.replace(regex substr, newSubStr function[, flags])</code> <code>рядок.replace(старий_фрагмент, нвоий_фрагмент)</code>	<pre>let hello = "Добрий день"; hello = hello.replace("день", "вечір"); // Добрий вечір</pre>
Заміна всіх входжень старого фрагменту тексту на новий фрагмент	<code>рядок.replaceAll(старий_фрагмент, нвоий_фрагмент)</code>	<pre>let s = "Там був Іван. Іван пив чай"; menu = s.replaceAll("Іван", "Петро"); // let menu = "Там був Петро. Петро пив чай";</pre>

Задача. Користувач вводить прізвище та ім'я в одному рядку через пробіл. Замінити пробіл на дефіс.

Задача. Користувач вводить дату у форматі «день.місяць.рік». Отримати рядкове представлення дати у форматі «день/місяць/рік»

## Видалення пробілів

		Приклад
Вилучення пробілів з початку і кінця рядка	<ul style="list-style-type: none"><li>• <b>trim()</b>: видаляє пробіли на початку і в кінці</li><li>• <b>trimStart()</b>: видаляє пробіли на початку</li><li>• <b>trimLeft()</b>:видаляє пробіли зліва</li><li>• <b>trimEnd()</b>:видаляє пробіли в кінці</li><li>• <b>trimRight()</b>: видаляє пробіли справа</li></ul>	<pre>let hello = "  Hello world  " let hello2 = hello.trim(); //"Hello world"  let hello3 = hello.trimStart(); //"Hello world  " let hello4 = hello.trimEnd (); //"  Hello world"</pre>



## Об'єднання, розділення рядків

Розділення рядка (формується масив з елементів між символами-розділювачами)	<code>рядок . split( символ_розділювач )</code>	<pre>let message = "Іван пив чай"; let stringArray = message.split(" "); // ['Іван', 'пив', 'чай'] for(var str in stringArray)     console.log(stringArray[str]);</pre>
формування рядка шляхом об'єднання елементів масиву та використання символу- розділювача (або фрагменту тексту)	<code>масив . join( символ_розділювач )</code>	<pre>let s = [1, 2, 3].join('-'); //1-2-3 &gt; 'aaax-bbx-bccc'.split('x-b') ↵ (3) ['aaa', 'b', 'ccc']</pre>

Задача. Користувач вводить прізвище, ім'я, та по-батькові через пробіл. Отримати рядок згідно формату «ім'я ПРІЗВИЩЕ» .

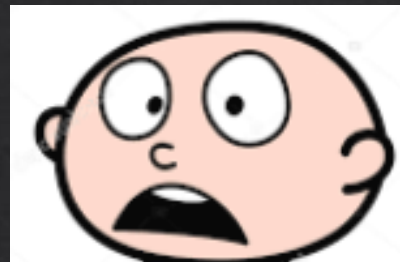
Задача. Дано масив цін. Сформувати 2 окремих рядки з цін розділених через кому які більші за 1000 і менші за 1000.

# Як розв'язати такі задачі ?

Приклад. Підрахувати кількість цифр у рядку

Приклад. Підрахувати кількість телефонних номерів у тексті

Приклад. Вибрати усі номери автомобілів з тексту



## Регулярні вирази

Регулярні вирази є об'єктами типу **RegExp**. Вони дозволяють описувати шаблони фрагментів тексту.

### Задання регулярного виразу

літерал	<code>/ шаблон / модифікатори ;</code>	<pre>let myExp = /hello/;  let r2= /^[xyz]/gi;  let myExp = new RegExp("hello");  let r2 = new RegExp("[xyz]", "gi")</pre>
створення як об'єкта	<code>new RegExp("шаблон", "модифікатори")</code>	

### Модифікатори

модифікатор	Загальна форма	Приклад
<b>i</b> – ігнорування регістру	<code>/ регулярний_вираз / i</code>	<pre>var str = "My Schools"; var patt1 = /schools/i; var result = str.match(patt1);</pre>
<b>g</b> – знаходження усіх фрагментів, які відповідають шаблону (пошук не зупиниться після знаходження першого фрагменту)	<code>/ регулярний_вираз / g</code>	<pre>var str = "Is this all there is?"; var patt1 = /is/g; var result = str.match(patt1); document.write(result);  &gt; result &lt; (2) ["is", "is"]</pre>
<b>m</b> – пошук у багаторядковому тексті	<code>/ регулярний_вираз / m</code>	<pre>var str = "\nIs th\nis it?"; var patt1 = /^is/m; var result = str.match(patt1);</pre>

[https://developer.mozilla.org/ru/docs/Web/JavaScript/Guide/Regular\\_expressions](https://developer.mozilla.org/ru/docs/Web/JavaScript/Guide/Regular_expressions)

Допустимими також є комбінації модифікаторів

<https://uk.javascript.info/regular-expressions>

## Деякі методи

	Загальна форма	Приклад
Перевірка рядка тексту на відповідність регулярному виразу (повертає true/false)	<code>регулярний вираз</code> . <b>test</b> ( <code>рядок тексту</code> )	<pre>var initialText = "hello world!"; var exp = /hello/; var result = exp.test(initialText); document.write(result + "&lt;br/&gt;"); // true  initialText = "beautifull wheather"; result = exp.test(initialText); document.write(result); // false - в строке initialText нет "hello"</pre>
Повертає частину рядка, яка відповідає регулярному виразу (null, якщо немає)	<code>регулярний вираз</code> . <b>exec</b> ( <code>рядок тексту</code> )	<pre>var initialText = "hello world!"; var exp = /hello/; var result = exp.exec(initialText); document.write(result + "&lt;br/&gt;"); // hello  initialText = "beautifull wheather"; result = exp.exec(initialText); document.write(result); // null //-----</pre>



## Деякі методи

Знаходження позиції першого входження фрагменту, що відповідає шаблону (регулярному виразу) у рядку (-1 якщо немає)	<code>рядок .search ( регулярний вираз )</code>	<pre>var str = "Visit My Schools" var n = str.search(/school/i) //9</pre>
Пошук усіх фрагментів, що відповідають заданому фрагменту (null, якщо немає) <u>додати модифікатор «g»!</u>	<code>рядок .match ( регулярний вираз )</code>	<pre>var initialText = "Дома ммаємо домен " var exp = /дом[a-я]*/gi; var result = initialText.match(exp); result.forEach(function(value, index, array){      document.write(value + "&lt;br/&gt;");  })</pre> <div>Дома домен</div>

## Параметри-списки (діапазони) символів

задаються за допомогою квадратних дужок		
<u>Один з вказаних символів</u> (у квадратних дужках перераховуємо допустимі символи)	[ <u>список символів</u> ]	<pre>let text = 'Номер мого авто AO1234BB. Мені 21 рік.' /[aouie]/.test(text) // ← повинен містити одну із букв // true  initialText = "город"; result = exp.test(initialText); document.write(result); // false</pre>
<u>Хоча би один не з вказаних символів</u> (перераховуємо заборонені символи і спочатку ставимо символ « ^ »)	[ ^ <u>список символів</u> ]	<pre>var str = "Do you know if this is all there is?"; var patt1 = /^[^is]/gi; var result = str.match(patt1); //D,o, ,y,o,u, ,k,n,o,w, ,f ,t,h, , ,a,l,l, ,t,h,e,r,e, ,?</pre>
<u>Символи з діапазону</u>	[ <u>start</u> - <u>end</u> ]	<pre>let text = 'Номер мого авто AO1234BB. Мені 21 рік.' /[a-я]/.test(text) // ← повинен містити одну із букв // true</pre>
<u>Не лише символи з діапазону</u>	[ ^ <u>start</u> - <u>end</u> ]	<pre>let text = 'Номер мого авто AO1234BB. Мені 21 рік.' /[^a-я]/.test(text) // ← не лише символи з діапазону  initialText = "3di0789"; exp = /^[^0-9]/; // ← не лише цифри result = exp.test(initialText); document.write(result); // true</pre>

Задача. Користувач вводить текст. Визначити, чи є там цифри і отримати список усіх цифр.

<u>альтернативні варіанти</u> (варіанти перераховуються у круглих дужках, розділені символами «   »)	( <span>варіант1</span>   <span>варіант2</span>   ...)	<pre>var str = "re, green, red, green, gren, gr, blue, yellow"; var patt1 = /(red green)/g; var result = str.match(patt1);     //green, red, green //-----      var str = "01234567890123456789";     var patt1 = /(0 5 7)/g;     var result = str.match(patt1);  //0,5,7,0,5,7</pre>
---	--	---

Задача. Користувач вводить день «sun,mon,tue,wed,thu,fri,sat»). Визначити, чи є це день вихідним

Задача. Рядко тексту, у якому згадуються різні моделі автомобілів.

«Іван Opel купив , а Оленка Audi. Їх товариш купив Mercedes. На підприємстві ж є Toyota, Nissan та Volkswagen»

Визначити чи є «BMW» там або «Audi». Та сформувати список усіх таких авто.

Задача. Дано перелік адрес ресурсів.

“ftp://test.test, http://sometest.test, https://sometest.test”

Перевірити, чи є там адреси сайтів (починаються з “http” або “https”).



# Метасимволи

	Приклад
. – довільний символ за винятком символу кінця рядка або нового рядка	<pre>var str = "That's hot!"; var patt1 = /h.t/g; var result = str.match(patt1);    //hat,hot</pre>
\d – довільна цифра (0-9)	<pre>let text = 'Номер мого авто A01234BB. Мені 21 рік.' text.match(/\d/g)    // ['1', '2', '3', '4', '2', '1']</pre>
\D -- не цифра	<pre>let text = 'Номер мого авто A01234BB. Мені 21 рік.' text.match(/\D/g)    // ['H','o','m','e','p',' ','m','o','r','o',' ','a','v','t','o',' ','A','O','B','B','.',',','M','e','h','i',' ','p','i','k','.']</pre>
\w – буква (латинського алфавіту), цифра, символ нижнього підкреслювання (A-Z, a-z, 0-9)	<pre>let h='I have 5001 friends !' h.match(/\w/g) //['I', 'h', 'a', 'v', 'e', '5', '0', '0', '1', 'f', 'r', 'i', 'e', 'n', 'd', 's'] //===== let text = 'Номер мого авто A01234BB. Мені 21 рік.' text.match(/\w/g)    // ['1', '2', '3', '4', '2', '1']</pre>
\W – не буква, не цифра, не символ нижнього підкреслювання	<pre>let h='I have 5001 friends !' h.match(/\W/g)    // [' ', ' ', ' ', ' ', ' ', ' '] //===== let text = 'Номер мого авто A01234BB. Мені 21 рік.' text.match(/\W/g) // ['H', 'o', 'm', 'e', 'p', ' ', 'm', 'o', 'r', 'o', ' ', 'a', 'v', 't', 'o', ' ', 'A', 'O', 'B', 'B', '.', ', ', 'M', 'e', 'h', 'i', ' ', 'p', 'i', 'k', '.']</pre>
\s – пробільний символ	<pre>let h='I have 5001 friends !' h.match(/\s/g) // [' ', ' ', ' ', ' ', ' ']</pre>
\S – не пробільний символ	<pre>let h='I have 5001 friends !' h.match(/\S/g)    // ['I','h','a','v','e','5','0','0','1','f','r','i','e','n','d','s','!']</pre>
\n – символ нового рядка	
\t – символ табуляції	

Квантифікатори

	Загальна форма	Приклад
+ -- принаймні одне входження (попереднього символу) кількість >= 1	<div>СИМВОЛ</div> +	<pre>var str = "Hellooo World! Hello W3Schools!"; var patt1 = /o+/g; var result = str.match(patt1); //ooo,o,o,oo</pre>
* -- довільна кількість або відсутність (попереднього символу) кількість >= 0	<div>СИМВОЛ</div> *	<pre>var str = "Hellooo World! Hello W3Schools!"; var patt1 = /lo*/g; var result = str.match(patt1); //l,looo,l,l,lo,l</pre>
? – входить один раз або відсутність  0<= кількість <= 1	<div>СИМВОЛ</div> ?	<pre>var str = "1, 100 or 1000?"; var patt1 = /10?/g; ar result = str.match(patt1); //1,10,10</pre>
рівно n символів  кількість = n	{n}	<pre>var str = "100, 1000 or 10000?"; var patt1 = /\d{4}/g; var result = str.match(patt1); //1000,1000</pre>
не менше n символів  кількість >= n	{n,}	<pre>var str = "100, 1000 or 10000?"; var patt1 = /\d{3,}/g; //100,1000,10000</pre>
від n до m  n <= кількість <= m	{n, m}	<pre>var str = "100, 1000 or 10000?"; var patt1 = /\d{3,4}/g; //100,1000,1000</pre>

```
/\d{1,2}\\/\d{1,2}\\/\d{4}/g  
12 / 11 / 2001
```

Задача. Дано повідомлення від користувача. Визначити, чи є всередині (не на початку і не у кінці) повідомлення рік, що починається з «19».

Задача. Дано повідомлення від користувача. Визначити, чи є у середині (не на початку і не у кінці) повідомлення номер телефону форматі «0-800-\*\*\*-\*\*\*» (приклад: 0-800-500-415).

## Позиція фрагмента пошуку (початок, кінець)

	Загальна форма	Приклад
початок рядка « ^ »		<pre>var str = "Is this his"; var patt1 = /^Is/g; var result = str.match(patt1); //Is</pre>
кінець рядка « \$ »		<pre>var str = "Is this his"; var patt1 = /is\$/g; //is  var str = "Is this his"; var patt1 = /\w*is\$/g; //his</pre>
початок або кінець слова « \b »		<pre>var str = "Visit My Schools"; var patt1 = /\bMy/g; var result = str.match(patt1); // My</pre>
не початок або кінець слова « \B »		<pre>var str = "Visit My Schools"; var patt1 = /\Bsit/g; var result = str.match(patt1); // sit</pre>




Задача. Визначити чи може бути рядок тексту номером банківської картки (приклад: «4142-3433-2323-3434»)

Задача. Дано адресу сайту. Визначити, чи є він урядовим (містить домен “gov”)

Задача. Вибрати усі роки після 2021 року з отриманого повідомлення (чотири цифри)



# Робота з групами (дозволяють групувати окремі фрагменти рядка)

Опис	Загальна форма	Приклад
групування, запам'ятовування знайдених фрагментів (якщо вказуємо у круглих дужках) (для доступу використовується: \$номер) Нумерація від 1	/...(group_pattern1) ... (group_pattern2)... /	<pre>var re = /(\\w+)\\s(\\w+)/i; var str = "Mikhail Bulgakov"; document.write(str.replace(re, "\$2, \$1")) // Bulgakov, Mikhail</pre>
Отримання груп	<div> <b>Позначення фрагментів</b>            \$n – n-това група            \$` - перед знайденим фрагментом            \$' - після знайденого фрагмента            \$+ - останній знайдений фрагмент            \$&amp; - знайдений фрагмент            \$_ - текст, де шукаємо            \$\$ - символ долара \$         </div>	<pre>const exp = /(\\d{4})-(\\d{2})-(\\d{2})/; const result = exp.exec("2021-09-06");</pre>  <pre>//===== const result3 = "2021-09-06".match(exp); result3 &gt; (4) ['2021-09-06', '2021', '09', '06', index: 0, input: '2021-09-06', groups: undefined]</pre>
Вираз може містити правила, що ґрунтуються на значенні деякої іншої групи	\номер_збереженої_групи	<pre>var str = "He said: \"She's the one!\"."; var reg = /(\\'1)(.*?)\\1/g; //Закривача лапка як і відкриваюча alert( str.match(reg) ); // "She's the one"</pre>
Іменовані групи	(?<назва_групи> ... )	<pre>Отримати значення атрибутів input let s = '&lt;input type="number" value="7"&gt;' let exp = /&lt;input type="(\\e1Type\\.*)" value="(\\&lt;val&gt;\\.*)"&gt;/ let res = exp.exec(s)  res.groups//об'єкт зі значеннями груп // {e1Type: 'number', val: '7'}</pre>
Групи не запам'ятовуються, використовуються для групування частин	(?: pattern)	<pre>/ма(?:ма ти)/ - короткий запис /мама мати/.</pre>

Позначення	Призначення	Приклад (умова + фрагмент з регулярок)
<code>\$n</code>	n-та група захоплення (capture group)	<code>"2025-10-06".replace(/(\d{4})-(\d{2})-(\d{2})/, "\$3.\$2.\$1")</code> → <code>"06.10.2025"</code>
<code>\$`</code>	текст перед знайденим фрагментом	<code>"HelloWorld".replace(/World/, "[\$`]")</code> → <code>"Hello[Hello]"</code>
<code>\$'</code>	текст після знайденого фрагмента	<code>"HelloWorld".replace(/Hello/, "[\$']")</code> → <code>"[World]World"</code>
<code>\$+</code>	остання знайдена група	<code>"abc123".replace(/(\d)(\d)(\d)/, "last=\$+")</code> → <code>"abc last=3"</code>
<code>\$&amp;</code>	сам знайдений фрагмент	<code>"foo123bar".replace(/\d+/, "[\$&amp;]")</code> → <code>"foo[123]bar"</code>
<code>\$_</code>	увесь вхідний рядок (де виконується пошук)	<code>"test123".replace(/\d+/, "[\$_]")</code> → <code>"test[test123]"</code>
<code>\$\$</code>	символ долара <code>\$</code>	<code>"price".replace(/price/, "\$\$100")</code> → <code>"\$100"</code>

Задачі.

- \$n Замінити всі дати формату DD-MM-YYYY на формат YYYY/MM/DD у тексті.
- \$& Обгорнути всі знайдені слова “JavaScript” у тексті в HTML-теги <strong>.
- \$` Порахувати кількість символів перед кожним знайденим словом “error” у лог-файлі.
- \$' Вивести текст, що йде після першого знайденого символу : у рядку.
- \$+ Замінити тільки останню групу у складному патерні на інший текст (наприклад, номер місяця у даті).
- \$\_ Відобразити повний рядок у повідомленні про помилку, коли знайдено певний патерн.
- \$\$ Замінити символ \$ на подвійний символ \$\$ у фінансовому тексті.

\$n – n-та група (нумерація з 1).  
\$& – весь знайдений фрагмент.  
\$` – текст перед знайденим фрагментом.  
\$' – текст після знайденого фрагмента.  
\$+ – остання знайдена група.  
\$\_ – увесь вихідний рядок.  
\$\$ – символ \$ .

### *Робота з групами (позиція фрагменту у залежності від розрашування інших фрагментів)*

якщо до слідує вказаний фрагмент	<code>(?&lt;= <u>фрагмент</u>)</code>	Отримати список значень всіх елементів нумерованого списку <code>let s="&lt;ol&gt;&lt;li&gt;11&lt;/li&gt;&lt;li&gt;22&lt;/li&gt;&lt;li&gt;33&lt;/li&gt;&lt;/ol&gt;"</code> <code>s.match(/(<u>?&lt;=&lt;li&gt;</u>)(.*?)(<u>?=&lt;\/li&gt;</u>)/g)</code>  <code>// ['11', '22', '33']</code>
якщо до не слідує вказаний фрагмент	<code>(?&lt;! <u>фрагмент</u>)</code>	
якщо після слідує вказаний фрагмент	<code>(?= <u>фрагмент</u> )</code>	<code>var str = "Is this all there is";</code> <code>var patt1 = /...is(=? <b>all</b>)/;</code> <code>var result = str.match(patt1);</code> <code>//this</code>
якщо після не слідує вказаний фрагмент	<code>(?! <u>фрагмент</u> )</code>	<code>var str = "Is this all there is";</code> <code>var patt1 = /is(?! <b>all</b>)/gi;</code> <code>var result = str.match(patt1);</code> <code>//Is,is</code>



Задача. Дано рядкове представлення дати у форматі «день.місяць.рік». Створити рядок у форматі «місяць/день/рік».

Задача. Дано рядок програми, яка містить звертання до властивості об'єкта за іменем (у лапках). Визначити чи правильно закрита лапка.

Задача. Дано рядок тексту, що містить фрагмент розмітки з таблицею.

```
“<table>  
  <tr>  
    <td>aa</td>  
    <td>bb</td>  
    <td>cc</td>  
  </tr>  
</table>
```

”

Сформувати список значень елементів (контенту, що міститься у середині тегу «td»)



## Розділення/заміна у рядках за регулярними виразами

Розділення рядка за шаблоном фрагмента-розділювача (не обов'язково одного символу)	<code>рядок . split ( регулярний вираз )</code>	<pre>var initialText = "Мама купила хліб"; var exp = /\s/; var result = initialText.split(exp); result.forEach(function(value, index, array){      document.write(value + "&lt;br/&gt;"); })  Мама купила Хліб</pre>
Заміна значень заданих шаблоном на новий фрагмент тексту	<code>рядок.replace(регулярний вираз, новий фрагмент)</code>	<pre>var re = /нудне/gi; var str = 'Програмування дуже нудне'; var newstr = str.replace(re, 'ціаве'); console.log(newstr); //Програмування дуже ціаве</pre>
Заміна рядка на основі функції формування нового рядка	<code>рядок.replace(регулярний вираз, функція)</code>	<pre>function replacer(match, p1, p2, p3, offset, string) {     // p1 - не цифри, p2 - цифри, p3 - не букви и     не цифри     return [p1, p2, p3].join(' - '); } var newString = 'abc12345#\$*%'.replace(/([^\d]*) (\d*) ([^\w]*)/, replacer); //abc - 12345 - #\$*%</pre>
Заміна кожного знайденого фрагменту (необхідно використати <code>\$&amp;</code> - знайдений фрагмент)	<code>рядок.replace(регулярний вираз, '... \$&amp; ...')</code>	<p>Урядку усі числа взяти в дужки</p> <pre>'some 343 text 34'.replace(/(\d+)/g, '(\$&amp;)')  //'some (343) text (34)'</pre>

Задача. Дано повідомлення. Отримати окремі речення (речення закінчується якимось розділовим знаком)

Задача. У тексті замінити усі «грн.» та «грн » на «grn»

Задача. Дано номер кредитної картки. Розділити кожні 4 цифри через пробіл.

*Жадібний і лінійний пошук (квантифікатор ?)*

«Жадібний» пошук  
(вибір максимально можливої кількості (перехід до наступного символу максимальну допустиму кількість раз) символів, які підходять за шаблоном)

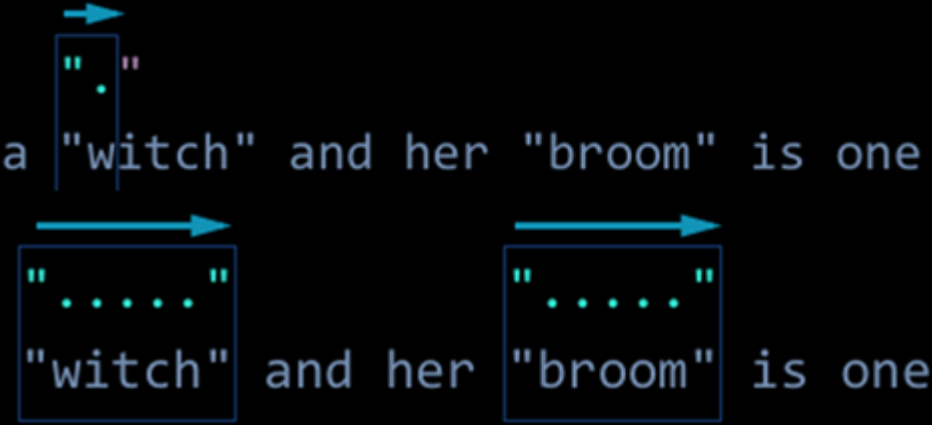
```
var reg = /".+"/g;  
var str = 'a "witch" and her "broom" is one';  
alert( str.match(reg) ); // "witch" and her  
"broom"
```



a "witch" and her "broom" is one

«лінійний» пошук  
(перехід до наступного символу згідно з шаблоном мінімальну можливу кількість )  
(для цього ставимо знак «?» після квантифікатора:  
\*? , +? , ??)

```
var reg = /".+?"/g;  
var str = 'a "witch" and her "broom" is one';  
alert( str.match(reg) ); // witch, broom
```



a "witch" and her "broom" is one

Задача. Отримати усі елементи тегів, що знаходяться у трикутних дужках “<....>”

«<ol>      <li>11</li>      <li>22</li>      <li>33</li>    </ol>»

