Number and String

План

- Задання числа
- Перевірка на число
- Округлення чисел
- Обєкт Math
- Випадкові числа
- об’єкт Date
- Задання строки
- Довжина
- Доступ до символів
- Пошук позиції підстроки
- Копіювання підстроки

Запис числа

`315` - ціле

`3.15` - дробове

`3.15e2` - з плаваючою точкою 3.15*100=315

`3.15e-2` - якщо відємний степінь 3.15*0.01=0.0315

`0xFF` - шістнадцятковий запис //255

Infinity

- математична безкінечність

```
5155/0                  // Infinity

-8458/0                 // -Infinity

0/0                     // NaN

Infinity>3155644984646              // true
```

NaN

- Not a Number (не число)
- Якщо не можливо виконати дію то результат рівний NaN
- NaN не рівна ніякому значенню включаючи себе
- NaN == NaN // false

```
10/‘10px’     // NaN

10+NaN                // NaN
```

Приведення до числа

- parseInt() - приводить строку до числа, якщо вона починається на число
- parseFloat() - приводить до числа символ за символом, поки це можливо

```
parseInt(‘25mm’);                    // 25

parseFloat(‘25.2mm’);        // 25.2

parseFloat(‘25.2.3’);                // 25.2

parseFloat(true);                    // NaN
```

object Number

- число можна визначити як об’єкт через new Number();
- typeof повертає об’єкт
- можна використовувати для приведення до числа

- Number(false); // 0
- Number('100a') // NaN

Округлення чисел

- Math.floor - округлює вниз
- Math.ceil - округлює вгору
- Math.round - округлює до найближчого цілого
- Math.trunc - відкидає дробову частинку

	`Math.floor`	`Math.ceil`	`Math.round`	`Math.trunc`
`3.1`	`3`	`4`	`3`	`3`
`3.6`	`3`	`4`	`4`	`3`
`-1.1`	`-2`	`-1`	`-1`	`-1`
`-1.6`	`-2`	`-1`	`-2`	`-1`

Округлення до заданої точності

```
var n = 26.1234;

n.toFixed(1);                    // 26.1

n.toFixed(5);                    // 26.12340 додає нулі в кінці

Math.round(n * 100) / 100;  // 26.1234 - 2612.34 - 2612 - 26.12
```

object Math

- об’єкт з математичними функціями

https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/Math

Випадкові числа (random)

- Math.random() - повертає випадкове число між 0 і 1

Math.floor(Math.random() * 10) + 1; - випадкове число від 1 до 10

Загальна формула випадкового числа:

```
Math.floor(Math.random() * (max - min + 1) ) + min;
```

Практика

- функція приймає межі випадкового числа (min, max) і вивести в консоль парне чи непарне воно

Date

- для роботи з датою і часом в JavaScript використовуються об'єкти Date

об’єкт з поточною датою

```
let now = new Date();
alert( now );
```

Об'єкт Date, значення якого дорівнює кількості мілісекунд (1/1000 секунди), що пройшли з 1 січня 1970 року GMT + 0.

https://learn.javascript.ru/datetime

Задання дати

```
new Date(year, month, date, hours, minutes, seconds, ms)
```

```
new Date(2011, 0, 1, 0, 0, 0, 0); // // 1 січень 2011, 00:00:00
```

Отримання дати

getFullYear ()	Отримати рік як чотиризначне число (yyyy)
getMonth ()	Отримати місяць як число (0-11)
getDate ()	Отримати день як число (1-31)
getHours ()	Отримати годину (0-23)
getMinutes ()	Отримати хвилину (0-59)
getSeconds ()	Отримати секунду (0-59)
getMilliseconds ()	Отримати мілісекунду (0-999)
getTime ()	Отримайте час (мілісекунди з 1 січня 1970 року)
getDay ()	Отримати робочий день як число (0-6)

https://www.w3schools.com/js/js_date_methods.asp

Задання дати

setDate	() Встановити день як число (1-31)
setFullYear	() Встановити рік (необов'язково місяць і день)
setHours	() Встановити годину (0-23)
setMilliseconds	() Встановити мілісекунди (0-999)
setMinutes	() Встановити хвилини (0-59)
setMonth	() Встановити місяць (0-11)
setSeconds	() Встановити секунди (0-59)
setTime	() Встановити час (мілісекунди з 1 січня 1970 року)

https://www.w3schools.com/js/js_date_methods_set.asp

Задання строки

Варіанти задання строки

```
let str = "Привіт";

let str2 = 'Одинарні лапки';

let phrase = `Зворотні лапки`;
```

Вставка змінної в строку

```
let price = 100;

let strPrice = 'Ваша ціна' + price + 'грн.';

let strPrice = `Ваша ціна ${price} грн.`;
```

В першому варіанті конкатинуємо строку з число, у випадку із зворотніми лапками вставляємо змінну в будь-яке місце строки через ${...}

Екранування

- якщо потрібно вставити в строку спецсимвол то ставиться екранування через \

  I\'m a JavaScript programmer';

  "I'm a JavaScript \"programmer\" ";

object String

- строку можна визначити як об’єкт через new String();
- typeof повертає об’єкт
- можна використовувати для приведення до строки

- String(10); // ‘10’

Довжина строки length

```
var str = ‘I am string’;

str.length;              // 11
```

Доступ до символів

- щоб отримати елемент використовується charAt(позиція)
- через квадратні дужки [позиція]

```
var str = ‘Some string’;
str.charAt(0);        // ‘S’
str[0];                          // ‘S’
"" .charAt (0);       // порожня строку
"" [0];                          // undefined
```

- str.charCodeAt(символ) - поверне unicode символа

Строки незмінні

Строку в JavaScript не можна змінити. Не можна взяти символ посередині і замінити його. Як тільки строка створена - вона така назавжди.

```
let str = 'Hi';

str[0] = 'h'; // error
```

Зміна регістру

- str.toUpperCase() - робить в строці всі великі літери
- str.toLowerCase() - робить в строці всі прописні літери

```
var str = "stringify";

str.toUpperCase(); // "STRINGIFY",
```

Позиція підстроки в строці indexOf

- поверне позицію на якій знаходиться підстрока
- якщо не знайдено то поверне -1
- другий параметр вказує на позиція з якої почати пошук, str.indexOf(‘sds’, position);

```
var str = "Please locate where locate occurs!";

var pos = str.indexOf("locate");
```

- є аналогічний метод lastIndexOf який шукає з кінця строки

includes()

- перевіряє, чи містить рядок задану підрядок, і повертає, відповідно true або false

```
var str = "Please locate where locate occurs!";

var pos = str.includes("locate");                    // true
```

startsWith endsWith

Методи **str.startsWith** і **str.endsWith** перевіряють, відповідно, починається чи і закінчується рядок певної рядком:

```
alert( "Widget".startsWith("Wid") ); // true, "Wid" — початок

alert( "Widget".endsWith("get") ); // true, "get" — закінчення
```

trim()

- видаляє пробіли з двох боків строки

```
var str = "          Hello World!          ";

str.trim(); // “Hello World!”
```

Копіювання підстроки

- `substring(start [, end])` повертає підстроку з позиції start до не включаючи end

```
var str= "stringify";

str.substring(0,1); // "s",
```

- `substr(start [, length])` повертає підстроку з позиції start, другий параметр кількість символів

```
str.substr(1,4); // "trin",
```

- `slice(start [, end])` повертає підстроку з позиції start до не включаючи end

```
var str = "stringify";

str.substring(0,1); // "s",
```

- відмінність в тому що slice може працювати з від’ємними значеннями які відраховуються від кінця строки

Порівняння строк

- строки порівнюються в алфавітному порядку і посимвольно
- порівнюється код в кодуванні unicode
- малі літери більше великих

```
alert( 'a' > 'Z' ); // true
```

Позиція підстроки в строці search()

- поверне позицію на якій знаходиться підстрока
- якщо не знайдено то поверне -1
- підтримує регулярні вирази

str.search(‘substring’);

replace()

- Змінює підстроку на підстроку

```
str = "Please visit Microsoft!";

var n = str.replace("Microsoft", "W3Schools"); // Please visit W3Schools!
```

Практика

- є строка ‘lorem ipsum is simply dummy’, зробити нову строку, щоб перше слову було з великої літери
- функція приймає дві строки і повертає більшу з них

Регулярні вирази

це шаблони використовуються для зіставлення послідовностей символів з шаблоном. Ці шаблони використовуються в методах **exec** і **test** об'єкта RegExp, а також **match**, **replace**, **search**, і **split** об'єкта String.

Регулярні вирази в JS

- В JavaScript регулярні вирази реалізовані окремим об'єктом RegExp і інтегровані в методи рядків

```
var regexp = new RegExp("шаблон", "флаги");
```

короткий запис (частіше використовується)

```
var regexp = /шаблон/; // без флагів
var regexp = /шаблон/gmi; //  з флагами gmi
```

Флаги

i

Якщо цей прапор є, то “регулярка” шукає незалежно від регістру, тобто не розрізняє між A і a.

g

Якщо цей прапор є, то “регулярка” шукає всі збіги, інакше - тільки перше.

m

Багаторядковий режим.

Методи

- search() - пошук у рядку за вказаним регулярним виразом, повертає індекс на якому я співпадіння і тільки перше
- match() - шукає та повертає (якщо є) відповідності рядка до зазначеного регулярного виразу.
- test() - виконує пошук на збіг між регулярним виразом і заданим рядком. Повертає true або false.
- replace() - повертає новий рядок з деякими або всіма порівняннями з шаблоном, замінениими на замінник

https://learn.javascript.ru/regexp-methods

Приклад

- знайти кількість входження підстроки в строку

```
var str = "Я люблю JavaScript!"; // будемо шукати в цьому рядку

alert (str.search (/ЛЮ/)); // -1
alert (str.search (/ЛЮ/ i)); // 2
```

Класи символів

спеціальне позначення, під яке підходить будь-який символ з певного набору.

\d - цифри.
\D - не цифри.
\s - пробільні символи, перенесення рядка.
\S - все, крім \s.
\w - латиниця, цифри, підкреслення '_'.
\W - все, крім \w.
'.' - точка позначає будь-який символ, крім перенесення рядка.
Якщо хочеться пошукати саме поєднання **"\d"** або символ «точка», то його екранують зворотним слешем, ось так: **\.**

https://learn.javascript.ru/regexp-character-classes

Приклади

- знайти всі числа в номері телефону

```
var str = "+7 (903) -123-45-67";
var reg = /\d/g;
alert (str.match(reg)); // масив всіх збігів: 7,9,0,3,1,2,3,4,5,6,7
```

Діапазони

- Квадратні дужки можуть також містити діапазони символів.

Наприклад, **[a-z]** - довільний символ від **a** до **z**, **[а-я]** - довільний символ від **а** до **я**, **[0-5]** - цифра від **0** до **5**.

- Квадратні дужки, що починаються зі знака каретки: **[^ ...]** знаходять будь-який символ, крім зазначених

[^0-9] - будь-який символ, крім цифри, теж що **\D**.

```
alert( "width: 100px".match(/[0-9][0-9][0-9]px/g) );
```

Посилання

https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/String

https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/Number

https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/RegExp

https://developer.mozilla.org/ru/docs/Web/JavaScript/Guide/Regular_Expressions

Відео

https://www.youtube.com/watch?v=9hLkbhRs7jM

https://www.youtube.com/watch?v=CAXBO9jOXFA