Lesson 5

План заняття

- Примітивні типи.
- Посилальний тип.

Примітивні типи

- String;
- Number;
- BigInt;
- Boolean;
- null;
- undefined;
- Symbol;

String

У JS будь-які текстові дані є рядками. Не існує окремого типу «символ», який є у низці інших мов.

Внутрішній формат для рядків завжди UTF-16, незалежно від кодування сторінки.

Приклад: U+0000

0	0	0	0	0	0	0	0	← 0
0	0	0	0	0	0	0	1	← 1
0	0	0	0	0	0	1	0	← 2
0	0	0	0	0	1	0	1	← 3

0	0	0	0	0	0	0	0	← a
0	0	0	0	0	0	0	1	← b
0	0	0	0	0	0	1	0	← c
0	0	0	0	0	1	0	1	← d

hello => 8 5 12 12 15
7 15 15 4 => good

console.log(String.fromCharCode(63)); => ?

Кодування — набір символів, закодованих за допомогою чисел для відображення тексту в електронному вигляді.

String

Важливо

Вміст рядка JavaScript не можна змінити. Не можна взяти символ посередині та замінити його. Як тільки рядок створено — він такий назавжди.

Як так робиться що ми маємо доступ до методів?

Конструктор String() створює об’єкти String. При виклику як функції вона повертає примітивні значення типу String.

Процедура:

1. Строрили змінну у яку записали данні (строку);
2. Ми звертаємось до методу строки (Наприклад .toLowerCase());
3. Створюється об’єкт огортка String() у конструктор якого передається значення строки;
4. Викликається метод класу String який віконує якусь операцію (Наприклад перетворює усі літери до прописних);
5. Метод повертає нову строку;
6. Огортка видаляється та залишає нове значення (строки);

String методи

- length – довжина строки. Це не метод, це ключове слово. Writable = false, enumerable = false, configurable = false;
- .at(index) || [index] – Отримання символа строки за індексом;
- .toUpperCase() – зміна регістру, перетворити до заголовних літер;
- .toLowerCase() – зміна регістру, перетворити до прописних літер;
- .indexOf(substr[, pos]) – Він шукає підстроку substr в рядку str, починаючи з позиції pos, і повертає позицію, на якій розміщено співпадіння, або -1 при відсутності їх відсутності. Шукає з початку строки.
- .lastIndexOf(substr[, pos]) – такий самий як і indexOf але шукає з кінця.

Важливо

indexOf не дуже корисний у логічних конструкціях тому що може повернути 0

indexOf та lastIndexOf – олдскул, краще користуйтесь новими методами.

String методи

- .includes(searchString[, pos]) – перевіряє, містить чи задану підстроку рядка, і повертає, відповідно true чи false.
- .startsWith(searchString[, pos]) – допомагає визначити, починається чи строка з символами вказаних у скобках, повертаючи, відповідно, true або false.
- .endsWith(searchString[, pos]) – допомагає визначити, закінчується чи строка з символами вказаних у скобках, повертаючи, відповідно, true або false.
- .slice(start [, end]) – Отримаємо частину від start до (не включно) end. Якщо аргументу end немає, slice повертає символи до кінця рядка. Також для start/end можна ставити негативні значення. Це означає, що позицію визначено як задану кількість символів з кінця рядка.
- .substring(start [, end]) – Повертає частину рядка між start та end (не включаючи) end. Це майже те саме, що і slice, але можна задавати start більше end. Якщо start більше end, то метод substring спрацює так, якби аргументи були поміняні місцями. Негативні значення substring, на відміну slice, не підтримує, вони інтерпретуються як 0.
- .substr(start [, length]) – Повертає частину рядка від start довжини length. Значення першого аргументу може бути негативним, тоді позиція визначається з кінця. Його мють підтримувати тільки деякі браузери, тому використовуйте краще slice.

String методи

- .replace(regexp|substr, newSubStr|function[, flags]) – повертає новий рядок із деякими чи всіма зіставленнями із шаблоном, заміненими на замінник. Шаблон може бути рядком або регулярним виразом, а замінник може бути рядком або функцією, що викликається при кожному зіставленні.

Вжливо

Аргумент flags не працює в ядрі v8 (движок JavaScript у Chrome та NodeJs).

- .trim() – видаляє пробіли з початку і кінця рядка.
- .split([separator[, limit]]) – розбиває об'єкт String на масив рядків шляхом поділу рядка зазначеним підрядком.

Separator - вказує символи, які використовуються як роздільник усередині рядка. Параметр separator може бути як рядком, і регулярним виразом.

Limit - ціле число, що визначає обмеження кількості знайдених підрядків. Метод split() все одно розділяє рядок на кожному зіставленні з роздільником separator, але обрізає масив, що повертається так, щоб він містив не більше limit елементів.

- .replace(regexp|substr, newSubStr|function[, flags]) -

Number

Об'єкт Number є обгорткою, що дозволяє вам працювати з числовими значеннями. Об'єкт Number створюється через конструктор Number().

- Якщо аргумент не може бути перетворений на число, повертається NaN.
- У неконструкторському контексті (тобто без оператора new), об'єкт Number може використовуватися для перетворення типів.

Властивості:

- Number.MAX_VALUE – приблизне значення = 1.79E+308;
- Number.MIN_VALUE – має значення приблизно дорівнює 5e-324. Значення, менші за MIN_VALUE, перетворюються на 0 (так зване «зникнення порядку» або «антипереповнення»);
- Number.NaN – представляє "не число". Еквівалентно глобальному об'єкту NaN;

Number

Методи:

- .isNaN() – визначає, чи є передане значення NaN. На відміну від глобальної функції isNaN(), Number.isNaN() немає проблеми примусового перетворення параметра в число. Це означає, що він безпечно передавати значення, які зазвичай перетворюються на NaN, але насправді NaN не є. Це означає, що метод повертає true тільки для числових значень, що мають значення NaN.
- .isFinite() – визначає, чи передане значення є кінцевим числом. На відміну від глобальної функції isFinite(), цей метод примусово не перетворює параметр на число. Це означає, що він повертає true тільки для кінцевих значень числового типу.
- .isInteger() – визначає, чи є передане значення цілим числом. Результат true або false;
- .isSafeInteger() – Визначає, чи передане значення є безпечним цілим числом (числом у діапазоні від -(253 - 1) до 253 - 1).
- .parseFloat(string) – являє собою той самий метод, що і метод parsefloat глобального об'єкта.
- .parseInt(string) – розбирає рядковий аргумент та повертає ціле число.

BigInt

BigInt це вбудований об'єкт, який надає спосіб репрезентувати цілі числа більше 253 - 1, найбільшого числа, яке JavaScript може надійно уявити з Number примітивом. Це максимальне значення можна отримати, звернувшись до Number.MAX_SAFE_INTEGER.

BigInt створюється шляхом додавання n в кінець цілого чисельного літералу — 10n — або викликом функції BigInt().

Наступні оператори можуть використовуватися з BigInt (або об'єктом-оберткою BigInt): +, *, -, **, %.

Унарний оператор, не підтримується (+)

BigInt дорівнює Number тільки за несуворого порівняння.

BigInt веде себе як звичайне число у таких випадках:

1. Перетворюється на Boolean через функцію Boolean
2. Використовується з логічними операторами Logical Operators (en-US) ||, && та !
3. В умовному тесті, такому як if statement.

Оскільки приведення між Number та BigInt може призвести до втрати точності, рекомендується використовувати BigInt тільки тоді, коли розумно очікуються значення, що перевищують 253 і не наводять між двома типами.

Boolean

Boolean([value]) є об'єктом-огорткою над примітивом логічного типу.

Значення, передане першим параметром, за потреби перетворюється на логічне значення. Якщо значення опущено або дорівнює 0, -0, null, false, NaN, undefined або порожній рядок (""), об'єкт має початкове значення, що дорівнює false. Всі інші значення, включаючи будь-які об'єкти або рядок "false", створюють об'єкт з початковим значенням, що дорівнює true.

Не плутайте примітивні значення true та false логічного типу зі значеннями true та false об'єкта Boolean.

Не використовуйте об'єкт Boolean для перетворення нелогічного значення на логічне значення. Замість цього використовуйте Boolean як функцію

Глобальний об'єкт Boolean не містить власних методів, однак він успадковує деякі методи з ланцюжка прототипів.

null

Не має глобальних об’єктів огорток

undefined

Не має глобальних об’єктів огорток

Symbol

Symbol([description]) – це унікальний та незмінний тип даних, який може бути використаний як ідентифікатор для властивостей об'єктів.

description – необов'язковий рядок. Опис символу, який можна використовувати під час налагодження, але не для доступу до самого символу.

Вбудовані символи:

- .iterator – метод, що повертає ітератор за промовчанням для об'єкта. Використовується конструкцією for...of;
- .match – метод для зіставлення об'єкта з рядком, також використовуваний визначення можливості об'єкта виступати як регулярного висловлювання;
- .replace – метод, що замінює збіглися підрядки в рядку. Використовується функцією String.prototype.replace();
- .search – метод, що повертає індекс входження підрядка, що відповідає регулярному виразу. Використовується функцією String.prototype.search();
- .split – Метод, що розбиває рядок на частини у місцях, що відповідають регулярному виразу. Використовується функцією String.prototype.split();

Symbol

Методи:

- Symbol.for(key) – шукає існуючі символи за заданим ключем і повертає його, якщо його знайдено. Інакше створюється новий символ для цього ключа у глобальному реєстрі символів.
- Symbol.keyFor(sym) – отримує за символом, що розділяється, його ключ із глобального реєстру символів.

Object

Конструктор Object створює об'єкт-огортку для переданого значення. Якщо значенням є null або undefined, створює і повертає порожній об'єкт, інакше повертає об'єкт такого типу, що відповідає переданому значенню. Якщо значення є об'єктом, конструктор поверне це значення.

При виклику в не-конструкторському контексті, Object веде себе ідентично до коду new Object().

Властивості:

- length – має значення 1;

Дякую за увагу