НЕДЕЛЯ 17 – СТРОКИ, ДАТЫ, Math

Вопросы 💎

1. Какие кавычки можно использовать для создания строк в JS и в чем разница между ними?

одинарные кавычки ';

двойные кавычки ";

обратный апостроф ` («шаблонная строка»).

Записи одинарными и двойными кавычками идентичны.

Шаблонная строка может быть многострочной, все переносы строк в ней будут сохранены.

В шаблонной строке с помощью синтаксиса ${ } можно использовать любые выражения JavaScript.

Например, для того, чтобы подставить в строку значения переменных между обратными апострофами пишется текст, а в местах, где нужно вставить значение из переменной используется синтаксис ${имя\_переменной}:

const name = "Таня";

const language = "JavaScript";

const weeks = 5;

Можно подставлять переменные:

`Меня зовут ${name}. Я изучаю ${language} уже ${weeks} недель. Но, кажется, ничего не понимаю.`;

Любой нестроковый результат (например, объект) будет приведён к строке.

Шаблонные строки сейчас — основной способ работы со строками, в которые нужно подставлять вычисляемые значения. Кроме этого, шаблонные строки закрывают недостатки обычных строк. Например, когда мы используем обычные строки, то символы кавычек внутри приходится экранировать обратным слэшем \. Чтобы JS понял, что на месте кавычек строка не заканчивается. С обратными кавычками это делать не нужно.

2. Какими методами можно найти подстроку в строке? Приведите пример.

Самый удобный способ получить подстроку — это метод substring.

Метод substring копирует указанную часть строки и возвращает копию в качестве результата. Метод принимает один или два аргумента. При вызове с двумя аргументами нужно передать индекс символа, с которого начинать копирование и индекс символа, на котором закончить. Индекс окончания не включается в копию.

const phrase = 'Пушкин'

const substring = phrase.substring(1, 4)

console.log(substring);

// ушк

console.log(phrase.substring(1, 5));

// ушки

Если указан только один аргумент, то результатом будет строка, начинающаяся с указанного индекса и до конца строки:

const phrase = 'Пушкин'

const substring = phrase.substring(0, 2)

console.log(substring);

// Пу

console.log(phrase.substring(1));

// ушкин

Существуют еще два похожих метода — substr и slice.

substr — устаревший метод, который будет удалён в будущих версиях языка, не нужно им пользоваться.

slice ведёт себя идентично substring, разница проявляется только если вызвать метод, поменяв местами индекс старта и индекс окончания копирования. В этом случае substring поймёт, что копировать, а slice вернёт undefined:

const phrase = 'Пушкин'

const substring = phrase.substring(0, 3)

console.log(substring);

// Пуш

console.log(phrase.slice(3, 0));

// undefined

Метод substring и slice часто используется в связке с indexOf — сначала находится индекс начала нужной подстроки, а затем этот индекс используется в substring как индекс начала копирования.

slice принимает отрицательные аргументы и удобен, когда нужно получить значение с конца строки. Например, частично скрывать длинный текст при отображении пользователю и показывать только первые и последние пять символов:

А для поиска одной строки внутри другой существуют следующие методы:

1️ - includes

Принимает аргументом строку, которую нужно найти.

Возвращает true, если строка нашлась, и false — если нет.

const phrase = 'Houston, we have a problem.'

console.log(phrase.includes('we'))

// true

2 - startsWith

Принимает аргументом строку, которую нужно найти.

Возвращает true, если текущая строка начинается с искомой и false — если нет.

const phrase = 'Hasta la vista, baby'

console.log(phrase.startsWith('Ha'));

// true

console.log(phrase.startsWith('la'));

// false

3️ - endsWith

Принимает аргументом строку, которую нужно найти.

Возвращает true, если текущая строка заканчивается искомой и false — если нет.

const phrase = 'I will think about that tomorrow'

console.log(phrase.endsWith('tomorrow'));

// true

console.log(phrase.endsWith('think'));

// false

4️ - indexOf

Принимает аргументом строку, которую нужно найти.

Возвращает индекс символа, с которого начинается искомая строка.

Если искомая строка не найдена, то возвращает -1.

const phrase = 'May the Force be with you'

console.log(phrase.indexOf('Force'));

// 8

console.log(phrase.indexOf('Jabba'));

// -1

Вторым аргументом методу можно передать индекс, с которого начинать поиск:

const phrase = 'May the Force be with you'

console.log(phrase.indexOf('r', 1));

// 10

3. Самостоятельно разберитесь, зачем нужен специальный символ `\n`?

Символ для форматирования текста при выводе на экран.

\n — начало новой строки;

\t — табуляция, аналогично нажатию кнопки Tab

Если эти символы есть в строке, то при печати на экран будут выполнены действия:

const mayakovsky = 'Я\n\tдостаю\n\t\tиз широких штанин\nдубликатом\n\t\tбесценного груза. \nЧитайте,\n\t\tзавидуйте,\n\t\t\t\tя —\n\t\t\t\t\tгражданин\nСоветского Союза.'

console.log(mayakovsky);

//Я

    достаю

        из широких штанин

дубликатом

        бесценного груза.

Читайте,

        завидуйте,

                я —

                   гражданин

Советского Союза.

4. Напишите код, который делает первый символ заглавным. Например, `"настя"` ⇒ `"Настя"`

let lowercasedName = 'настя';

let commonName = lowercasedName[0].toUpperCase() + lowercasedName.slice(1);

console.log(commonName);

//Настя

5. Как создать дату 24 января 2021 года, 22 часа 51 минута? Временная зона – местная.

let testData = new Date(2021, 0, 24, 22, 51);

console.log(testData);

//Sun Jan 24 2021 22:51:00 GMT+0300 (GMT+03:00)

6. Как посчитать, сколько секунд осталось до завтра?

let date = new Date();

console.log(date);

let tomorrow = new Date(2022, 6, 24);

console.log(tomorrow);

console.log(+tomorrow - +date);

// Sat Jul 23 2022 22:31:24 GMT+0300

// Sun Jul 24 2022 00:00:00 GMT+0300

// 5315110

7. Как выделить из строки с денежной суммой (например, \*120р.\* или \*99€\*) только цифры, т.е. \*120 и 99\*?

8. Зачем нужны функции `join` и `split`?

Метод split позволяет разбить строку на отдельные подстроки. Чаще всего это нужно, чтобы разбить строку на слова.

Метод принимает аргументом разделитель, по которому нужно делить строку на подстроки. Возвращает массив получившихся подстрок.

В практическом задании № 1 нужно было разбить введенное ФИО на отдельные поля Ф, И, О, разделение по пробелам:

const fullName = document.getElementById('hello-name');

const nameSplit = fullName.value.split(' ');

Наоборот, склеить массив строк в одну можно методом join, он принимает один аргумент — строку, которая будет использоваться для склейки строк.

Например, склеить строки пробелами:

[ 'I am', 'happier', 'than ever.' ].join(' ');

//'I am', 'happier', 'than ever.'

9. Какой из вариантов округления делает это по математическим правилам?

round — округление по обычным (математическим) правилам;

floor — округление вниз;

ceil — округление вверх;

let num = 13.87

// Обычное округление

console.log(Math.round(num)) // 14

// Округление до ближайшего целого в большую сторону

console.log(Math.ceil(num)) // 14

// Округление до ближайшего целого в меньшую сторону

console.log(Math.floor(num)) // 13

10. Как сгенерировать случайное число от 1 до 100?

let max = 1;

let min = 100;

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

11. Зачем нужна функция `str.trim()` ?

Для очистки строк от пробелов и символов окончания строки. Метод не принимает аргументов, а возвращает строку без пробелов в начале и конце строки:

const  parol = '    YpChM7#lz   '

console.log(parol.trim());

// 'YpChM7#lz'

12. Что такое флаг?

По умолчанию функция`replace()` заменяет только первое совпадение.

Чтобы заменить все совпадения, используйте регулярное выражение с `/g` флагом:

str = "Пожалуйста посетите веб-сайт Microsoft! Microsoft - известная американская компания.";

var n = str.replace(/Microsoft/g, "Google");

```

По умолчанию функция `replace()` учитывает регистр. Писать MICROSOFT (большими буквами) не получится.

Для замены без учета регистра используйте регулярное выражение с `/i` флагом:

str = "Пожалуйста, посетите веб-сайт Microsoft!";

var n = str.replace(/MICROSOFT/i, "Google");