Объект Math

Объект Math

Объект Math является встроенным объектом, хранящим в своих свойствах и методах различные **математические константы и функции**. Объект Math не является функциональным объектом.

Math не работает с числами типа **BigInt**.

В отличие от других глобальных объектов, объект Math **не является конструктором**. Все свойства и методы объекта Math являются **статическими**, поэтому вы не должны пытаться вызывать метод на созданном экземпляре объекта Math.

Написанное выше есть парадигма объектного программирования. Это сложно объяснить начинающему программисту. В уроке "Работа с числами" я приводил пример объектов и экземпляров. Повторю еще раз, с учетом сказанного про объект Math.

example_1. Статические методы объекта

```
<script>
    // --- строка (объект String) ---
    // инициализация строки через создание объекта String
    // str - экземпляр объекта String
    let str = new String("Welcome pechora_PRO ...");
    // вызов метода includes на экземпляре объекта String
```

```
    console.log(str.includes("pechora")); // true
    // --- число (объект Number) ---
    // инициализация числа через создание объекта Number
    // num - экземпляр объекта Number
    let num = new Number(2.333);
    // вызов метода toString на экземпляре объекта Number
    console.log(num.toString()); // '2.333'
    // вызов статического метода parseInt на объекте Number
    // без создания экземпляра Number
    console.log(Number.parseInt()); // 2
    // --- объект Math ---
    // вызов статического метода sqrt на объекте Math
    // без создания экземпляра Math
    console.log(Math.sqrt(25)); // 5
    // к статическим методам объекта необходимо обращаться
    // -> без создания экземпляра этого объекта
</script>
```

Даже если и сейчас смысл примера остался непонятен, ничего страшного, для этого мы и учимся. Всему свое время.

Свойства объекта Math

Приведу два примера **свойств** объекта **Math**. Может быть даже не для использования, но вы должны знать и до изучения объектов, - каждый объект имеет набор **свойств** и **методов**.

Math.PI

Свойство Math.**PI** представляет отношение длины окружности круга к его диаметру, приблизительно равное **3,14159**.

Поскольку свойство PI является **статическим свойством** объекта Math, вы всегда должны использовать его как Math.PI, а **не пытаться создавать экземпляр** объекта Math и получать свойство от него (объект Math не является конструктором).

Math.E

Свойство Math.**E** (число **Эйлера** или Непера) представляет основание натурального логарифма **e**, приблизительно равное **2,718**.

Свойство **E** является статическим свойством объекта Math.

example_2. Свойства объекта Math

```
<script>
    // числовая переменная
    // если число не введено - по умолчанию 5
    let r = prompt("Введите радиус окружности") || 5;
    // если введено "не совсем" число
    r = parseFloat(r);
    // расчет длины окружности
    let P = 2 * Math.PI * r;
    // расчет площади круга
    let S = Math.PI * r * r;
    // вывод в браузер расчетных данных
```

```
    document.write("<h2>При r = ", r, "</h2>");
    document.write("<h3>Длина окружности: ", P, "</h3>");
    document.write("<h3>Площадь круга: ", S, "</h3>");
</script>
```

Методы объекта Math

Math.floor

Метод Math.**floor**() - округление вниз. Округляет аргумент до ближайшего меньшего целого.

```
Math.floor(x);
```

Параметр

- **x** – число.

Метод **floor**() является статическим методом объекта Math.

Math.ceil

Метод Math.**ceil**() - округление вверх. Округляет аргумент до ближайшего большего целого.

```
Math.ceil(x);
```

Параметр

- **x** – число.

Метод **ceil**() является статическим методом объекта Math.

example_3. Тестирование методов ceil(), floor()

```
<script>
	// инициализация переменной
	let a = 11 / 3;
	// вывод исходного числа
	console.log("11 / 3 = ", a);
	// вывод округленных значений
	console.log(".ceil() - ", Math.ceil(a));
	console.log(".floor() - ", Math.floor(a));
</script>
```

Math.round

Метод Math.**round**() возвращает число, округлённое к ближайшему целому.

```
Math.round(x);
```

Параметр

- **x** – число.

Если дробная часть числа больше, либо равна 0,5, аргумент будет округлён до ближайшего большего целого. Если дробная часть числа меньше 0,5, аргумент будет округлён до ближайшего меньшего целого.

Метод **round**() является статическим методом объекта Math.

Math.trunc

Функция Math.**trunc**() возвращает целую часть числа путём удаления всех дробных знаков.

```
Math.trunc(x);
```

Параметр

- **x** – число.

Возвращаемое значение

Целая часть данного числа.

В отличие от других трёх методов объекта Math — Math.**floor**(), Math.**ceil**() и Math.**round**() — метод Math.**trunc**() работает очень просто. Отбрасывается запятая и все цифры после неё, не обращая внимания на знак аргумента.

Аргумент, переданный в этот метод, будет неявно преобразован в число.

Метод **trunc**() является статическим методом объекта Math.

example_4. Тестирование методов округления чисел

```
<script>
	// инициализация переменной
	a = 19 / 7;
	// вывод исходного числа
	console.log("19 / 7 = ", a);
	// вывод округленных значений
	console.log(".ceil() - ", Math.ceil(a));
	console.log(".floor() - ", Math.floor(a));
	console.log(".round() - ", Math.round(a));
	console.log(".trunc() - ", Math.trunc(a));
</script>
```

Math.pow

Метод Math.**pow**() возводит число в заданную степень. Первым параметром передается число, вторым - в какую степень его возвести.

```
Math.pow(base, exponent);
```

Параметры

- **base** – основание степени.
- **exponent** – показатель степени, в которую возводится основание **base**.

Метод **pow**() является статическим методом объекта Math.

example_5. Тестирование метода pow()

```
<script>
	// расчет площади круга
	let r = 5;
	let S = Math.PI * Math.pow(r, 2);
	// расчет дискриминанта квадратного уравнения
	let b = 8;
	let a = 2;
	let c = 1;
	let D = Math.pow(b,2) - 4 * a * c;
	console.log("Площадь круга: ", S);
	console.log("Дискриминант: ", D);
</script>
```

Math.sqrt

Метод Math.**sqrt**() возвращает квадратный корень числа.

```
Math.sqrt(x);
```

Параметр

- **x** – число.

Метод **sqrt**() является статическим методом объекта Math.

Возвращаемое значение

Квадратный корень заданного числа. Если число отрицательное, то вернётся **NaN**.

example_6. Тестирование метода sqrt()

```
<script>
	// расчет корней квадратного уравнения
	// инициализация коэффициентов
	let a = 1;
	let b = 6;
	let c = 4;
	// расчет дискриминанта
	let D = Math.pow(b,2) - 4 * a * c;
	// расчет корней уравнения
	let x1 = (-b + Math.sqrt(D)) / 2 * a;
	let x2 = (-b - Math.sqrt(D)) / 2 * a;
	// вывод расчетных значений
	console.log("Дискриминант: ", D);
	console.log("x1: ", x1);
	console.log("x2: ", x2);
</script>
```

Math.random

Метод Math.**random**() возвращает **псевдослучайное число с плавающей запятой** из диапазона [0, 1), то есть, от 0 (включительно) до 1 (но не включая 1), которое затем можно **отмасштабировать** до нужного диапазона.

```
Math.random();
```

Возвращаемое значение

Псевдослучайное число с плавающей запятой от 0 (включительно) до 1 (не считая).

> ***Важно***. Метод Math.random() не предоставляет криптографически стойкие случайные числа. Не используйте его ни для чего, связанного с безопасностью.

- **Получение случайного числа от 0 (включительно) до 1 (не включая)**

```
Math.random();
```

- **Получение случайного числа в заданном интервале**

Этот пример возвращает случайное **число** в заданном интервале. Возвращаемое значение не менее (и может быть равно) min и не более (и не равно) max.

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

example_7. Получение случайного числа в заданном интервале

```
<script>
    // выбираем изображение
    let bear = document.querySelector("#happy-bear");
    // ширину изображения для тестирования в консоль
    // console.log(bear.style.width);
    // диапазон размеров для атрибута width
    let min = 150;
    let max = 600;
    // получаем ширину из указанного диапазона
    let width = Math.random() * (max - min) + min;
```

```
    // новую ширину изображения для тестирования в консоль
    console.log(width);
    // устанавливаем новую ширину для изображения
    bear.style.width = width + "px";
</script>
```

- **Получение случайного целого числа в заданном интервале**

Этот пример возвращает случайное **целое число** в заданном интервале. Возвращаемое значение не менее min (или следующее целое число, которое больше min, если min не целое) и не более (но не равно) max.

```
min = Math.ceil(min);
max = Math.floor(max);
// максимум не включается, минимум включается
Math.floor(Math.random() * (max - min)) + min;
```

example_8. Получение случайного целого числа в заданном интервале

```
<script>
    // делаем выборку элементов p
    let el = document.querySelectorAll("p");
    // определяем диапазон
    let min = 0;
    let max = el.length;
    // console.log(max);
    min = Math.ceil(min);
    max = Math.floor(max);
    // получаем случайное число в диапазоне min - max
    let num = Math.floor(Math.random() * (max - min )) + min;
```

```
    // console.log(num);
    // получаем доступ к цитате
    citate = el[num].innerHTML;
    // выводим цитату в браузер
    document.write("<pre><h2>" + citate + "</h2></pre>");
</script>
```

- **Получение случайного целого числа в заданном интервале, включительно**

Если вам нужно, чтобы включалось и минимальное, и максимальное значение, можно использовать следующее решение.

```
min = Math.ceil(min);
max = Math.floor(max);
// максимум и минимум включаются
Math.floor(Math.random() * (max - min + 1)) + min;
```

example_9. Получение случайного целого числа в заданном интервале, включительно

```
<script>
    let min = 1;
    let max = 6;
    min = Math.ceil(min);
    max = Math.floor(max);
    // получаем случайное число
    // максимум и минимум включаются
    let num = Math.floor(Math.random() * (max - min + 1)) + min;
    // console.log(num);
    // формируем элемент изображения
```

```
    var img = `<img src='${num}.png'>`;
    // вывод сообщения
    alert("Вы набрали " + num + " баллов");
    // вывод изображения
    document.write(img);
</script>
```

Math.max / Math.min

Метод Math.**max**() возвращает наибольшее из нуля или более чисел.

Метод Math.**min**() возвращает наименьшее из нуля или более чисел.

```
Math.max([value1[, value2[, ...]]]);

Math.min([value1[, value2[, ...]]]);
```

Параметры

- value1, value2, ... - числа.

Возвращаемое значение

При вызове без аргументов результатом вызова будет значение **-Infinity**.

Если хотя бы один из аргументов не может быть преобразован в число, результатом будет **NaN**.

Методы **max**() / **min**() являются статическими методами объекта Math.

example_10. Тестирование методов max(), min()

```
<script>
    // диапазон для генератора случайных чисел
    let rangeMin = 0;
    let rangeMax = 50;
```

```
    // генерируем пять случайных чисел
    let num1 = Math.floor(Math.random() * (rangeMax - rangeMin))
    + rangeMin;
    let num2 = Math.floor(Math.random() * (rangeMax - rangeMin))
    + rangeMin;
    let num3 = Math.floor(Math.random() * (rangeMax - rangeMin))
    + rangeMin;
    let num4 = Math.floor(Math.random() * (rangeMax - rangeMin))
    + rangeMin;
    let num5 = Math.floor(Math.random() * (rangeMax - rangeMin))
    + rangeMin;
    // вывод в консоль
    console.log(num1, num2, num3, num4, num5);
    // из диапазона чисел выбираем
    // -> минимальное
    let min = Math.min(num1, num2, num3, num4, num5);
    // -> максимальное
    let max = Math.max(num1, num2, num3, num4, num5);
    // выводим найденные значения в консоль
    console.log("Min = ", min);
    console.log("Max = ", max);
</script>
```

Задание № 15

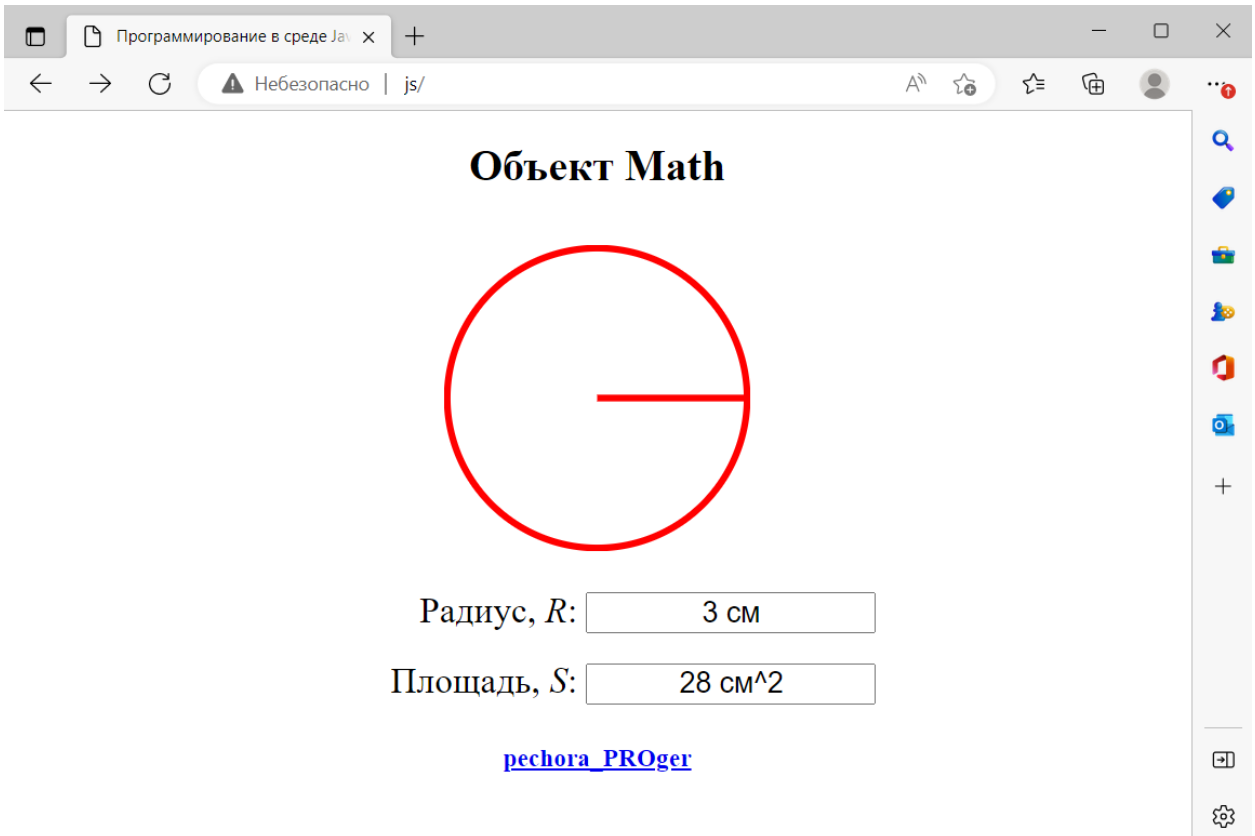
В раздаточном материале представлен скрипт, реализующий построение окружности с **радиусом**, вычисляемым **случайным** образом.

Примечание. Тег <**svg**> используется как контейнер для хранения SVG графики. SVG (Scalable Vector Graphics – масштабируемая векторная графика) - язык разметки, расширенный из XML для описания двухмерной векторной графики.

Вычислите **площадь** построенной **окружности**. В специальные текстовые поля запишите расчетные значения:

- **радиуса** окружности
- **площади** окружности

Примерный вывод результата работы скрипта представлен на рисунке.

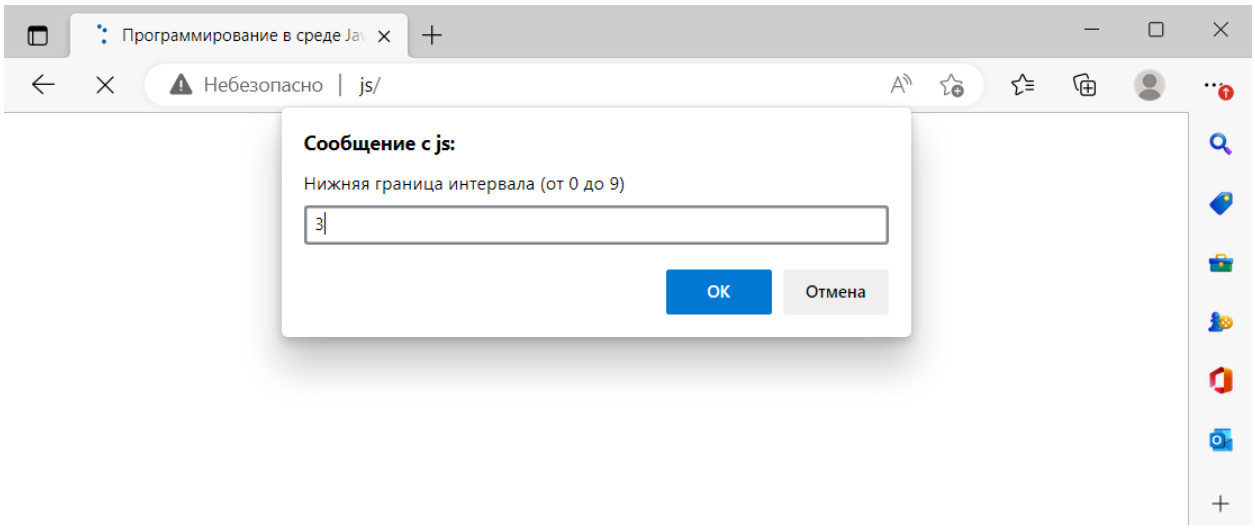


Задание № 16

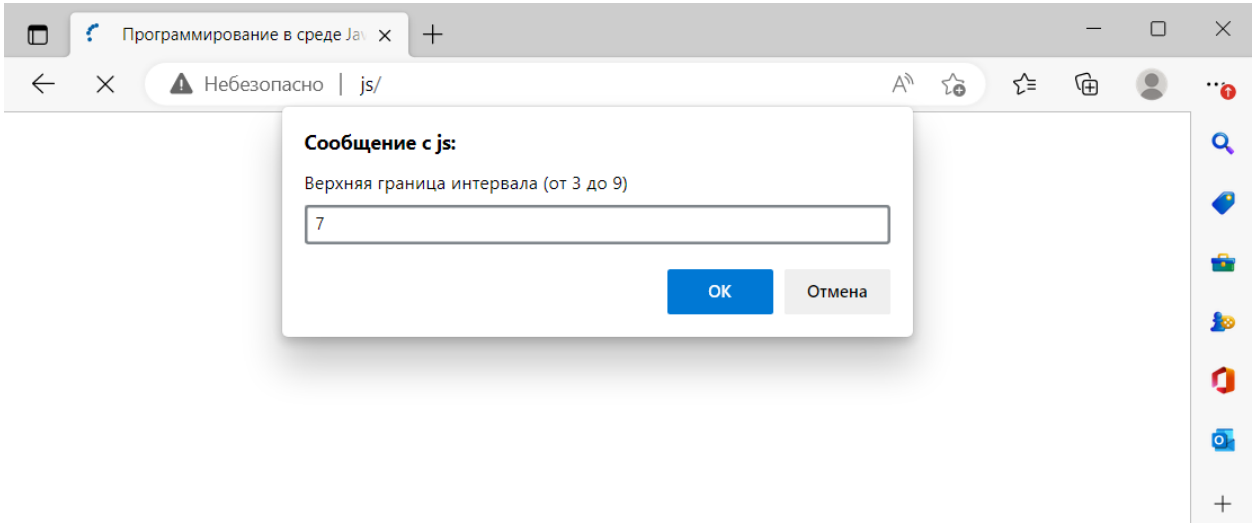
В файле **data.js** раздаточного материала представлен **JS-массив объектов**, каждый из которых представляет тизер фильма. Подключите файл массива к файлу **index.html**. Напишите **скрипт** вывода **случайного тизера**.

Для определения интервала поиска случайного числа используйте диалоговое окно метода **prompt**(). По умолчанию интервал находится в диапазоне индексов массива **data** (0-9).

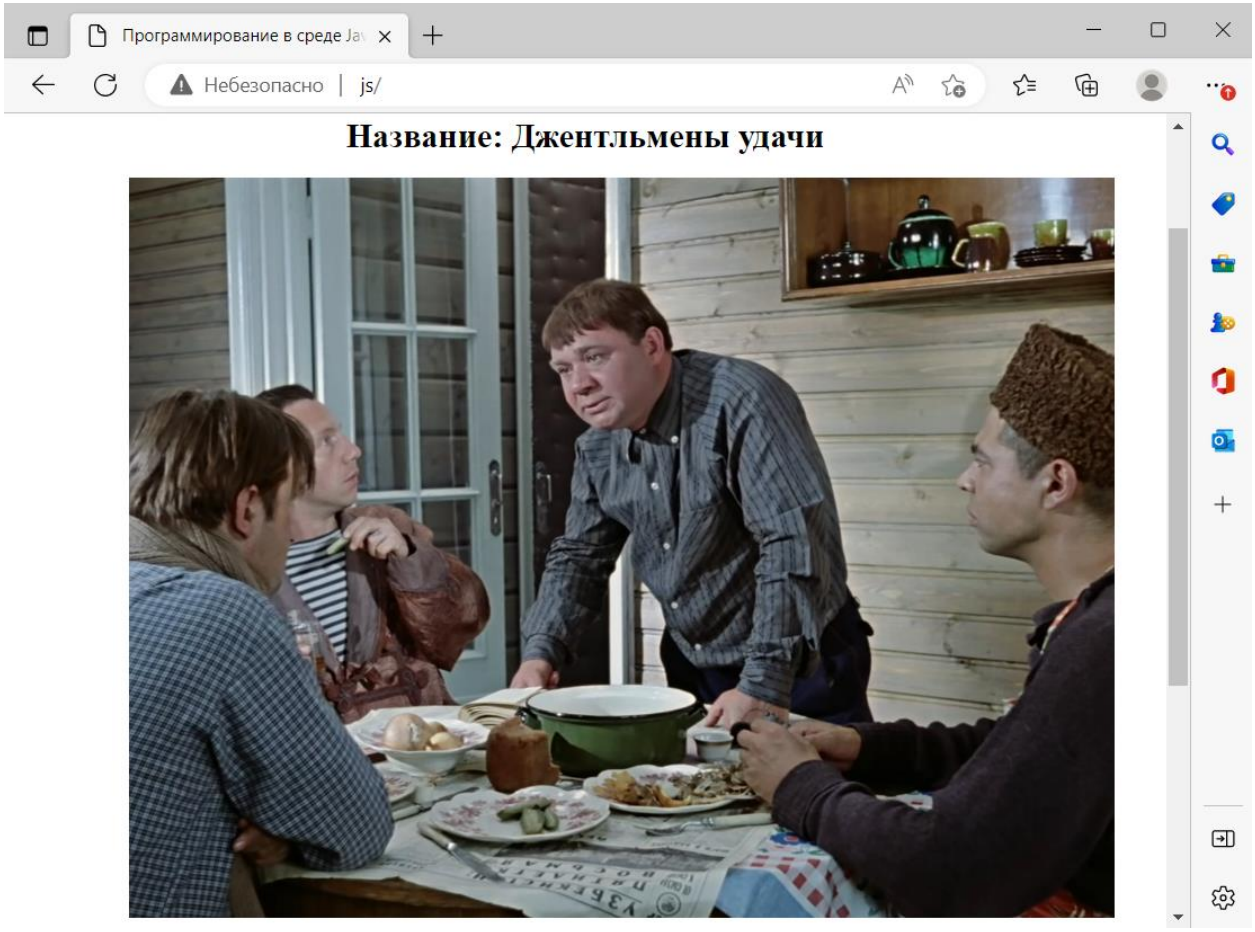
Запрос **нижней** границы интервала.



Запрос **верхней** границы интервала.



Примерный вывод тизера фильма.



Задание № 17

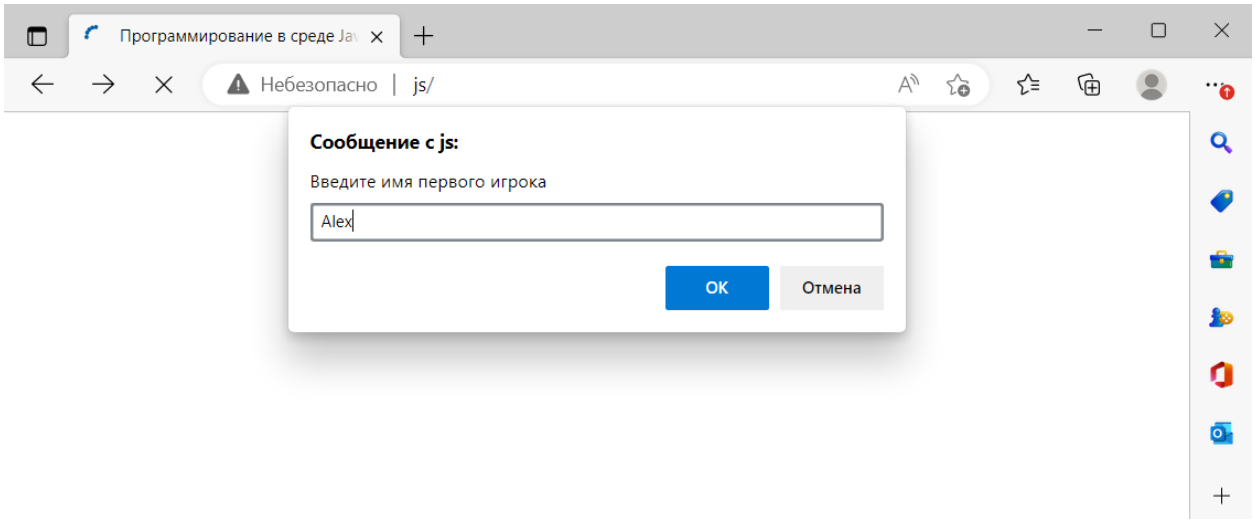
Напишите скрипт, имитирующий **бросок** игровых **кубиков** двумя игроками. Данные об игроках сохранить в объекте. В качестве данных сохраняем **имя** игрока и **количество набранных баллов** от броска трех кубиков.

```
// объект для записи данных игроков
let game = {
      gamer1: {
            name: "",
            count: undefined
      },
      gamer2: {
            name: "",
            count: undefined
      }
}
```

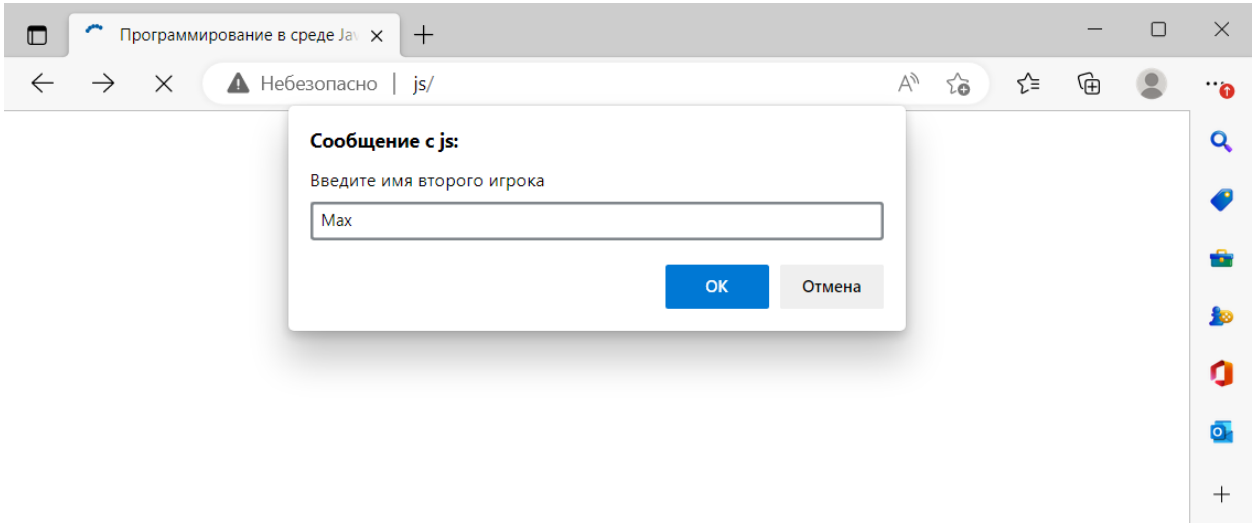
Загрузка изображения кубика определяется случайным числом, в диапазоне 1 – 6.

Имена игроков запросить с помощью диалогового окна метода **prompt**().

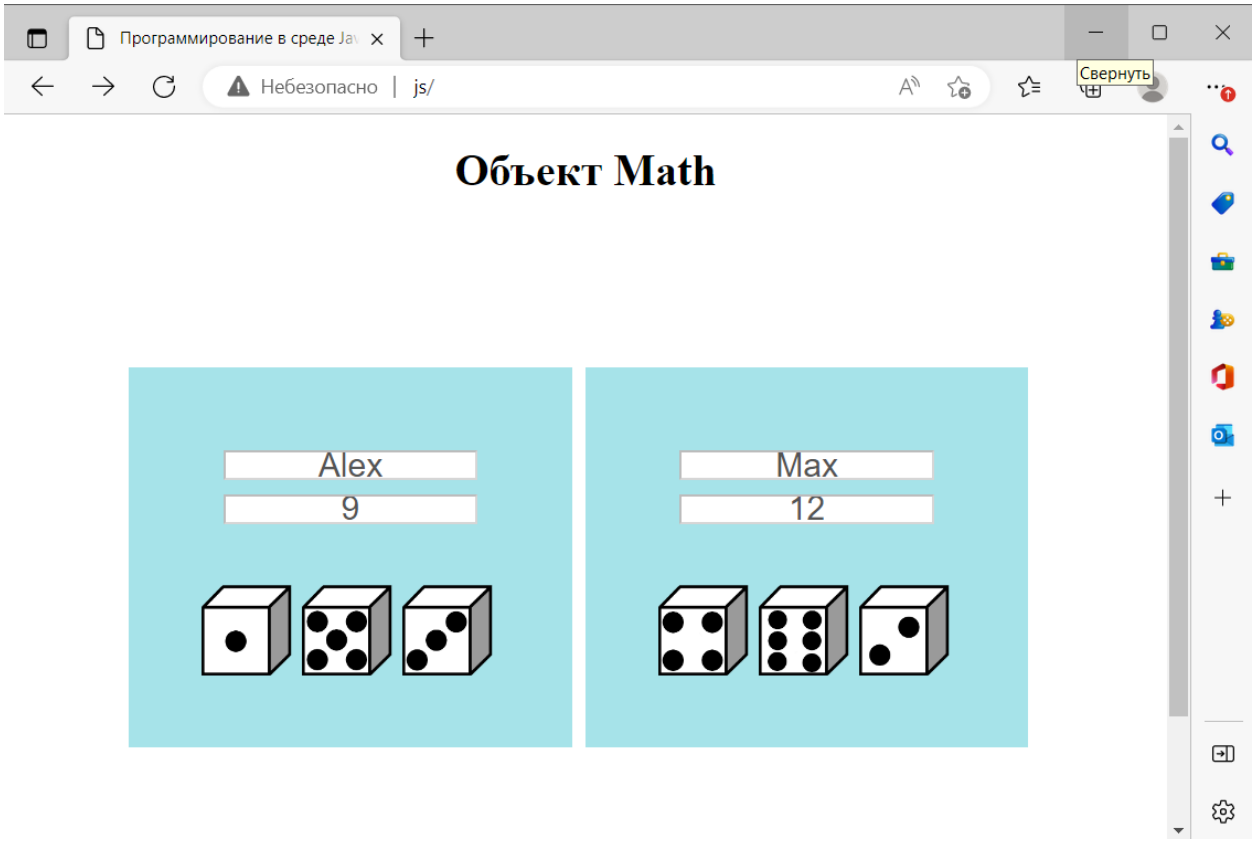
Запрос имени первого игрока.



Запрос имени второго игрока.



Результат броска вывести в соответствующие **текстовые поля**. Примерный **результат** работы скрипта представлен на рисунке.



Программа может иметь множество вариантов исполнения. Я предлагаю самый простой и очевидный способ, ваш вариант может (и должен) отличаться от предложенного.

Главное – движение. Конечная цель остается прежней, написание сценариев **клиент-серверного** взаимодействия.

Задание № 18

Напишите скрипт нахождения **корней квадратного уравнения** для следующих коэффициентов:

- **a = 2**
- **b = 9**
- **c = 5**

Коэффициенты ввести используя диалоговое окно метода **prompt**().

Общий вид квадратного уравнения:

- **$ax^2 + bx + c = 0$**

Дискриминант квадратного уравнения находится следующей формулой:

- **d = Math.pow(b, 2) - 4 * a * c**

Для предложенных коэффициентов уравнение имеет два корня:

- **x1 = (-b + Math.sqrt(d)) / (2 * a)**
- **x2 = (-b - Math.sqrt(d)) / (2 * a)**

Округлите найденные корни (x1, x2) до ближайшего целого.

Выведите расчетные значения дискриминанта и корней уравнения в **консоль**.