**Урок №2. Типы данных. Создание примитивных объектов на canvas**

**Вводный блок (5 минут)**

**Цели и задачи урока**

1. «Целью нашего занятия является знакомство с типами данных, которые используются при работе с алгоритмами».
2. «Рассмотрим основные примитивные фигуры, которые используются для отрисовки компонентов на холсте».

**Обсуждение**

Вопросы к ученикам:

1. Как вы думаете, что такое данные?
2. Какого вида данные бывают?

**Познание нового (10 минут)**

**Данные**

Данные - отображение информации в различных вариантах. Это могут быть числа, текст, цвет и так далее. И для того, чтобы правильно интерпретировать ту информацию, которая хранится в компьютере, используют соответствующие типы данных.

Сами типы данных делятся на примитивные и объекты. Примитивные типы данных хранят в себе что-то одно. Про объекты будем говорить позже.

**Типы данных**

На языке JavaScript основными примитивными типами данных являются:

* числа (number) - хранят в себе как целые числа, так и дробные числа;
* строки (string) - любой набор символов, заключенный в кавычки (одинарные или двойные);
* булева (boolean) - тип, который может хранить в себе только правду или ложь, то есть имеет всего два состояния.

**Некомпьютерная активность (10 минут)**

1. Показать несколько вариантов данных и спросить, к какому из типов данных относится представленный тип.

Например:

92 - число

3256.53 - число

35235ке3 - строка

ЛОЖЬ - булева

2. Игра в ассоциации. Каждый ученик получает карточку с названием типа данных (например, целые числа, вещественные числа, строки и т. д.). Задача участников — объяснить тип данных с помощью ассоциаций и примеров, не называя его напрямую. Остальные ученики должны угадать, о каком типе данных идет речь.

**Познание нового (10 минут)**

**Переменные**

Информацию где-то хранят, если ее необходимо использовать несколько раз в работе программы, для этого используют переменные.

Переменные - пространство в памяти, к которому можно получить доступ по его имени. При этом, переменные могут быть изменяемые, в ходе работы программы (**let**), и неизменяемые (**const**).

Далее представлен синтаксис создания переменных различных типов данных

let a = 144

let b = "Строка"

let c = true;

a = a \* 2;

**Вопрос к ученикам**

1. К какому типу данных относятся вышеперечисленные переменные?
2. Как вы думаете, что означает звездочка в последней строке?
3. Какие математические операции вы знаете?

**Математические операции**

«+» - сложение

«-» - вычитание

«\*» - умножение

«/» - деление

«\*\*» - возведение в степень

«%» - остаток от деления

**Некомпьютерная активность (10 минут)**

1. Решение примеров. Вывести примеры и попросить детей их решить.

let a = 15;

let b = (a + 10) \* 10

let c = b / (a - 5);

// Чему равен с?

let x = 5;

let y = 20 \* x \*\* 2 + 5 \* x + 100;

//Чему равен y?

1. Поделить детей на команды по 3-4 человека. Придумать по 2-3 примера для других команд.

**Работа над проектом (40 минут)**

Тема: Создание нового проекта и написание программы, которая позволяет выводить фигуры сложной формы на Canvas.

**Рисование примитивов**

Теперь попробуем создать несколько примитивных фигур на холсте.

**Вопрос ученикам:**

Знаете ли вы определение «Примитивные фигуры»?

**Ответ на вопрос:**

Примитивные фигуры - это фигуры, из которых состоят сложные геометрические фигуры, например - прямоугольник, круг и т.д.

Дальше мы с помощью холста создадим несколько примитивных фигур.

Откроем проект в Visual Studio Code и воспользуемся шаблоном с прошлого занятия

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Document</title>

</head>

<body>

<canvas width="700px" height="700px" style="border: 5px solid black;">

</canvas>

<script>

const canvas = document.querySelector('canvas');

const context = canvas.getContext('2d');

</script>

</body>

</html>

**Прямоугольники**

Начнем с прямоугольника. Для отрисовки прямоугольника используются две функции: fillRect и StrokeRect.

fillRect используется для закрашенного квадрата, в strokeRect закрашиваются только границы.

Пример:

<script>

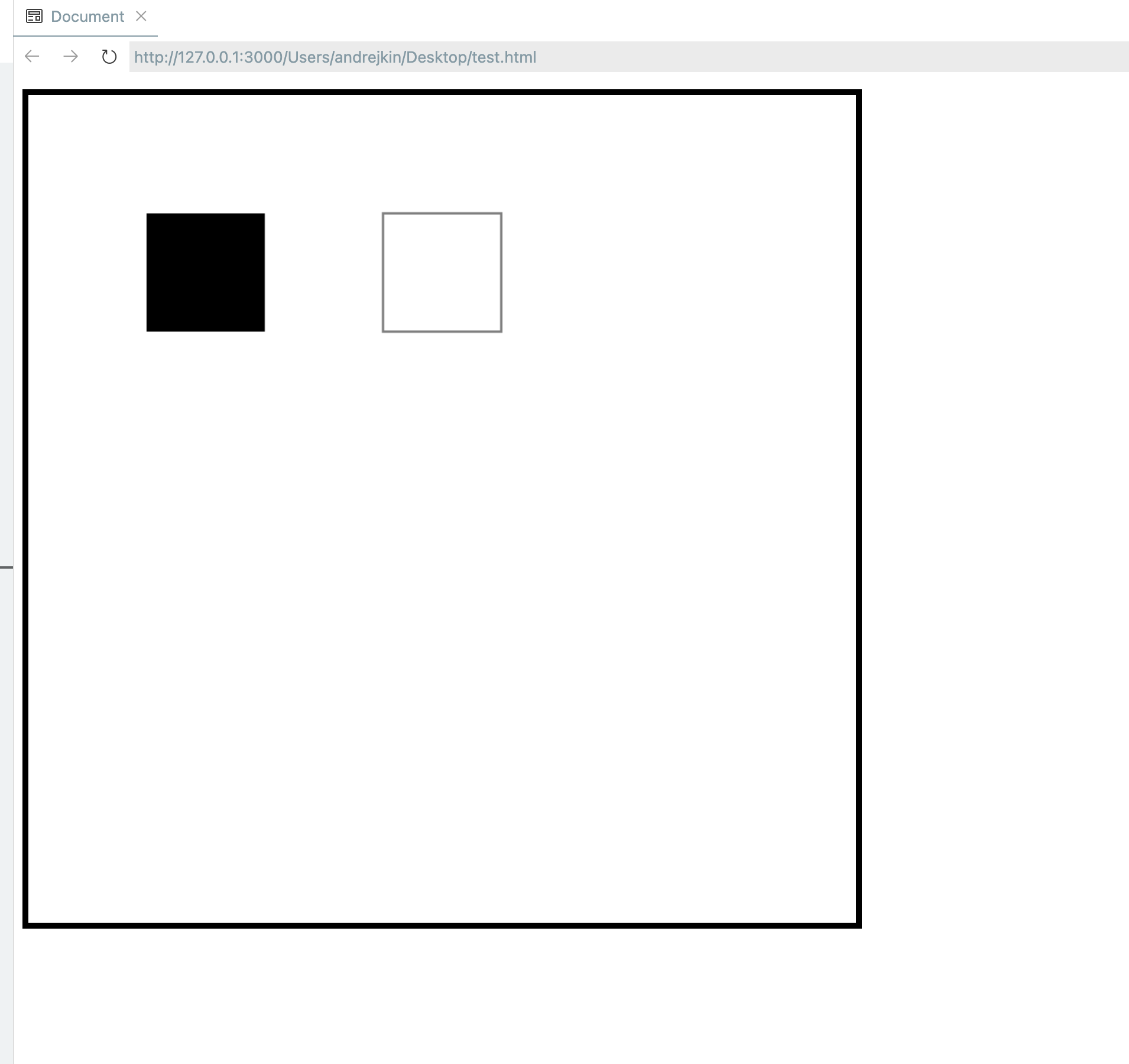
const canvas = document.querySelector('canvas');

const context = canvas.getContext('2d');

**context.fillRect(100,100,100,100);**

**context.strokeRect(300,100,100,100);**

</script>



**Вопрос ученикам:**

Какие используются типы данных в скобках?

**Ответ:**

Числа

Для изменения цвета используются команды fillStyle и strokeStyle.

<script>

const canvas = document.querySelector('canvas');

const context = canvas.getContext('2d');

**context.fillStyle = "rgb(180,100,0)";**

**context.strokeStyle = "rgb(0,180,100)";**

context.fillRect(100, 100, 100, 100);

context.strokeRect(300, 100, 100, 100);

</script>

**Вопрос ученикам:**

Какие используются типы данных в скобках?

**Ответ:**

Строки

**Задание для учеников:**

1. Создать закрашенный прямоугольник фиолетового цвета размерами 100х400 на координатах 300х200.

2. Создать незакрашенный прямоугольник с границей красного цвета размерами 100х75 на координатах 400х50.

**Линии**

Для отрисовки линий используются следующие команды:

**moveTo** - переместиться в определенную точку на холсте;

**lineTo** - отрисовать от точки, которая указывается в moveTo до точки, которая указана в данной команде;

**beginPath** - начало отрисовки;

**closePath** - окончание отрисовки;

**stroke** - построить фигуру.

Например, мы хотим отрисовать линию от координат 200х200 до координат 400х400

<script>

const canvas = document.querySelector('canvas');

const context = canvas.getContext('2d');

**context.beginPath();**

**context.moveTo(200, 200);**

**context.lineTo(400, 400);**

**context.stroke();**

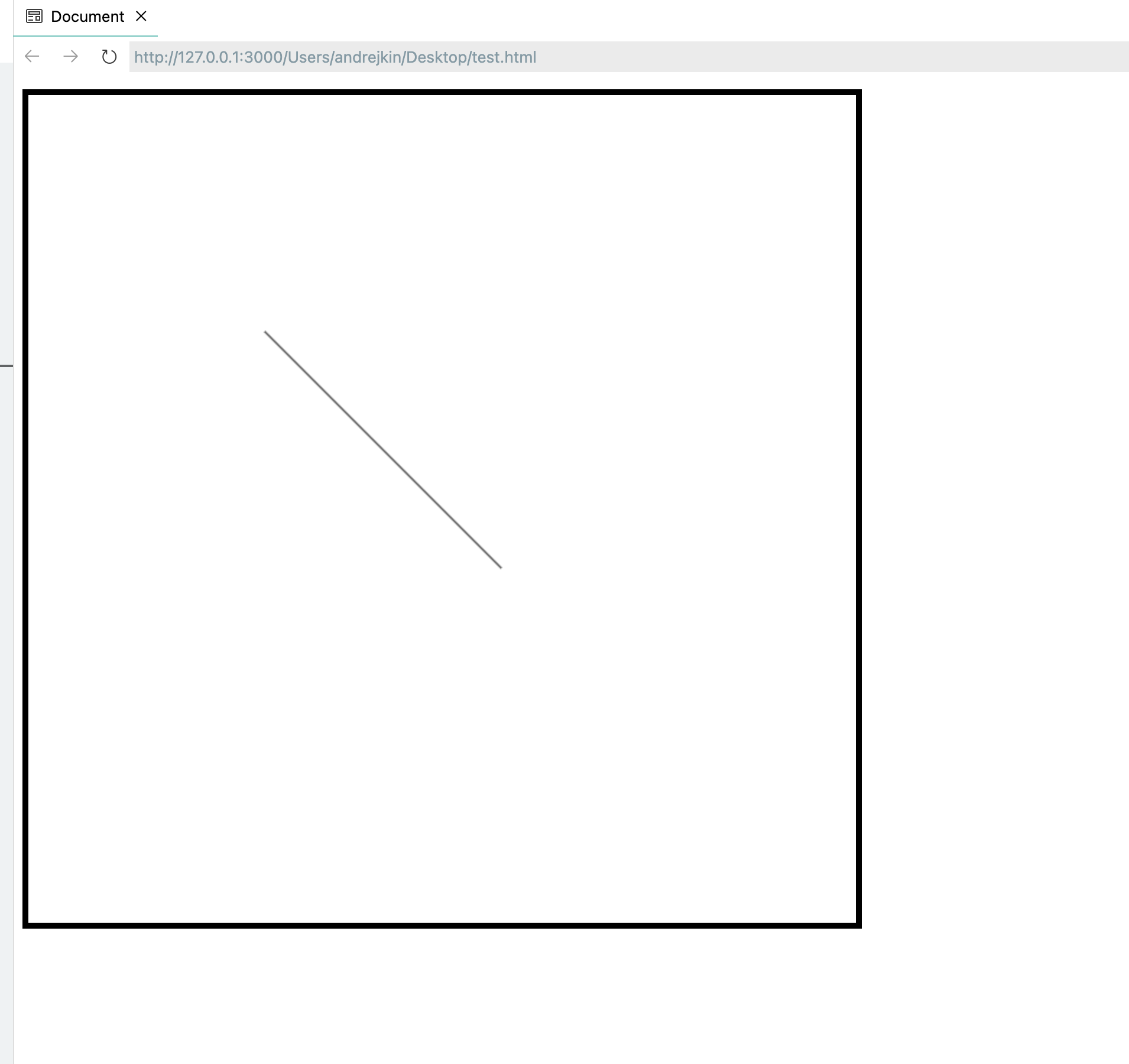
**context.closePath();**

</script>

Таким образом можно строить и более сложные фигуры (ромбы, треугольники и т.д.), достаточно использовать функцию lineTo несколько раз.

<script>

const canvas = document.querySelector('canvas');

const context = canvas.getContext('2d');

context.beginPath();

context.moveTo(200, 200);

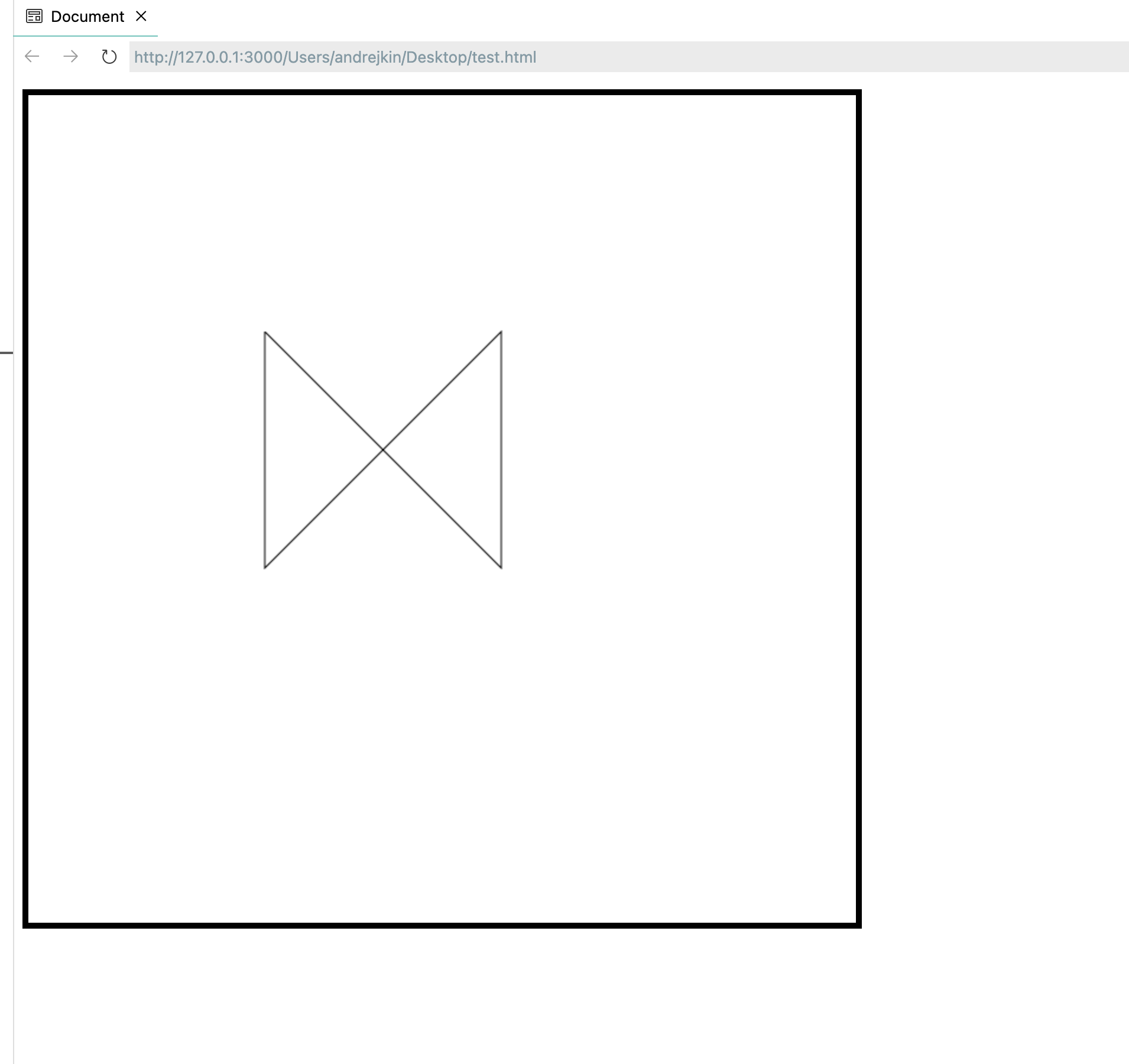
context.lineTo(400, 400);

context.lineTo(400, 200);

context.lineTo(200, 400);

context.lineTo(200, 200);

context.stroke();

</script>

**Задание для учеников:**

Создать треугольник любых размеров.

**Круги**

Для отрисовки круга или дуги используется команда arc

**context.arc(x, y, радиус, начальный\_угол, конечный угол).**

Начальный и конечный угол указываются в радианах. Для того, чтобы перейти к системе радиан, необходимо Число Пи разделить на 180 и умножить на нужный угол.

<script>

const canvas = document.querySelector('canvas');

const context = canvas.getContext('2d');

context.beginPath();

context.arc(200, 200, 100, 0, (Math.PI / 180) \* 360);

context.fill();

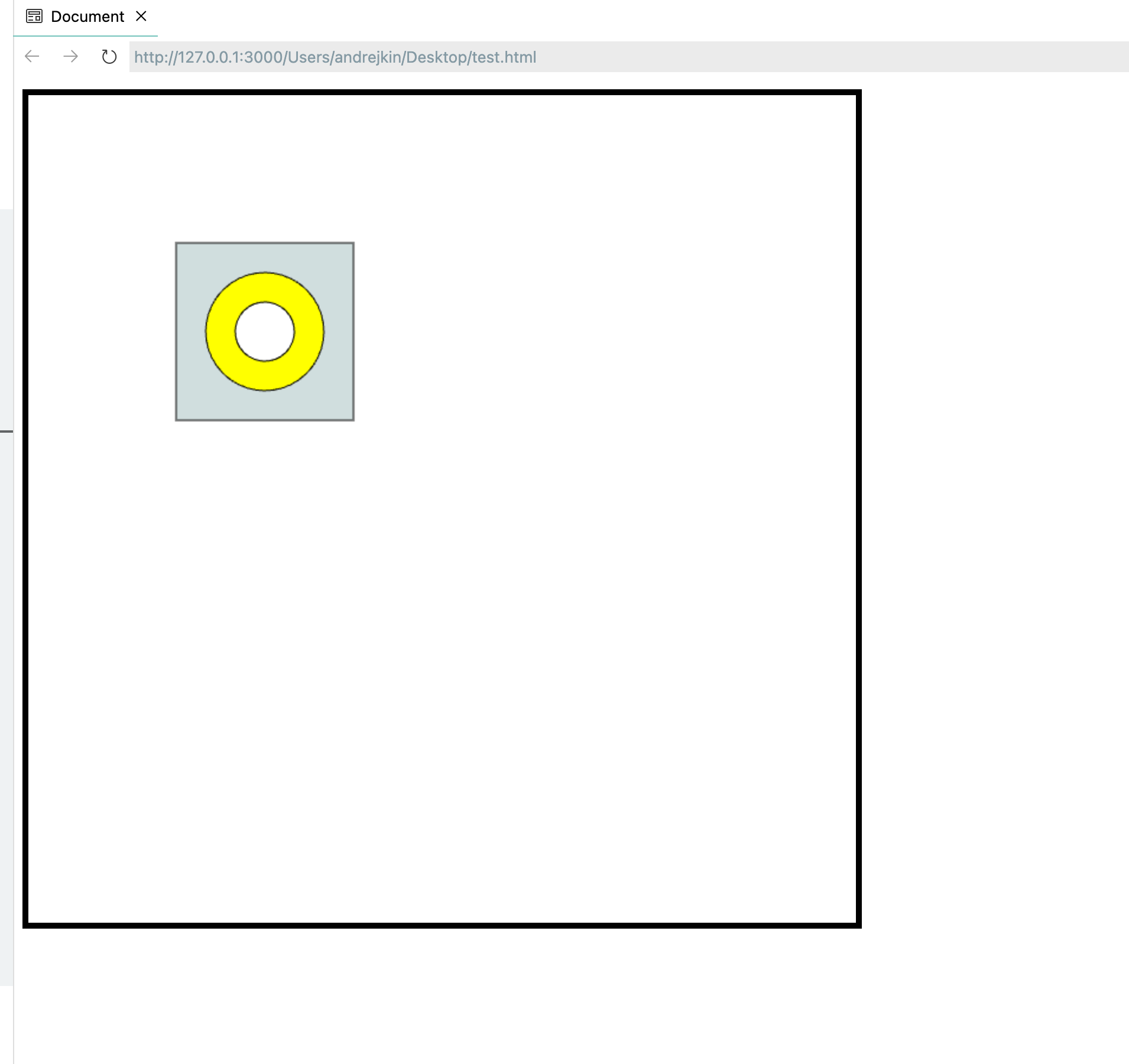
context.closePath();

</script>

В данном случае используется fill вместо stroke для того, чтобы вся внутренняя часть была закрашена.

**Задание для учеников:**

Нарисовать фигуру из нескольких объектов согласно примеру



**Дополнительное задание:**

Нарисовать сложный объект (например, дом или машину).

**Рефлексия (5 минут)**

"Что нового вы узнали сегодня?"

"Что такое данные? Какие типы данных бывают?"

"Какие примитивные фигуры вы сегодня рисовали? С помощью каких функций?"