Московский государственный технический
университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»
Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Парадигмы и конструкции языков программирования»

Отчет по домашней работе
«Tetris на JS»

Выполнил:	Проверил:
студент группы ИУ5-32Б	преподаватель каф. ИУ5
Гайнуллин А. М.	Гапанюк Ю.Е.

Описание задания

1. Выберите язык программирования (который Вы ранее не изучали) и (1) напишите по нему реферат с примерами кода или (2) реализуйте на нем небольшой проект (с детальным текстовым описанием).
2. Реферат (проект) может быть посвящен отдельному аспекту (аспектам) языка или содержать решение какой-либо задачи на этом языке.
3. Необходимо установить на свой компьютер компилятор (интерпретатор, транспилятор) этого языка и произвольную среду разработки.
4. В случае написания реферата необходимо разработать и откомпилировать примеры кода (или модифицировать стандартные примеры).
5. В случае создания проекта необходимо детально комментировать код.
6. При написании реферата (создании проекта) необходимо изучить и корректно использовать особенности парадигмы языка и основных конструкций данного языка.
7. Приветствуется написание черновика статьи по результатам выполнения ДЗ.
8. Черновик статьи может быть подготовлен группой студентов, которые исследовали один и тот же аспект в нескольких языках или решили одинаковую задачу на нескольких языках.

Текст программы

Файл index.html

```html
<!DOCTYPE html>
<html lang="en" dir="ltr">
 <head>
  <meta charset="utf-8" />
  <title>Tetris!</title>
  <script src="js/app.js" charset="utf-8"></script>
  <link rel="stylesheet" href="css/style.css" />
  <link
   href="https://fonts.googleapis.com/css?family=Montserrat:300,400&display=swap"
   rel="stylesheet"
  />
 </head>

 <body>
  <header>
   <h1 class="fw-300 t-ucase">
    <br /><span class="fw-400 t-wide t-big t-ucase">Cool tetris</span>
   </h1>
  </header>
  <main class="game-area">
   <div class="game">
    <div class="grid"></div>
    </div>
    <section>
      <div class="display">
        <h1 class="score fw-400 t-ucase">
          Your Score <br />
          <span class="score-display t-ucase fw-300">0</span>
        </h1>
        <div class="previous-shape">
          <div class="previous-grid"></div>
        </div>
        <h2 class="lines-display fw-400 t-ucase">
          Lines:<span class="lines-score">0</span>
        </h2>
      </div>
      <button class="button t-ucase" href="#">Start / Pause</button>
      <small class="footer t-ucase">By Artem Gainullin</small>
    </section>
  </main>
 </body>
</html>
```

Файл style.css

```css
/* Здесь все мои стили*/
:root {
  font-size: 0.625em;
}

* {
  box-sizing: border-box;
}

body {
  font-family: "Montserrat", sans-serif;
  font-size: 1.6rem;
  margin: auto;
  max-width: 60rem;
  color: #d8edea;
  background: radial-gradient(
    circle,
    rgba(175, 196, 174, 1) 0%,
    rgba(104, 204, 191, 1) 89%,
    rgba(94, 191, 178, 1) 100%
  );
}

header {
  text-align: center;
  margin-top: 3rem;
}

div {
  height: 2rem;
  width: 2rem;
}


.t-ucase {
  text-transform: uppercase;
}

.t-big {
  font-size: 1.5em;
}

.t-wide {
  letter-spacing: 1.5rem;
}

.t-close {
  letter-spacing: 1rem;
}

.fw-300 {
  font-weight: 300;
}

.fw-400 {
  font-weight: 400;
}
```

```css
.score-display {
  font-size: 5rem;
  color: rgb(133, 121, 107, 0.5);
}

.game-area {
  display: flex;
  justify-content: center;
}

.game {
  height: 0;
  width: 300px;
}

.grid {
  display: flex;
  flex-wrap: wrap;
  align-items: center;
  width: 20rem;
  height: 40rem;
}

.previous-shape {
  width: 10rem;
  padding-left: 2rem;
  margin-top: -5rem;
}

.previous-grid {
  display: flex;
  flex-wrap: wrap;
  width: 8rem;
  height: 8rem;
}

.block {
  background-image: url(../images/blue_block.png);
}
```

```css
.block2 {
  background-image: url(../images/purple_block.png);
}

.block3 {
  background-image: url(../images/green_block.png);
}

.block4 {
  background-image: url(../images/navy_block.png);
}

.block5 {
  background-image: url(../images/pink_block.png);
}

.end {
  background-color: #d8edea;
}

.button {
  position: relative;
  width: 22rem;
  height: 2.2rem;
  text-align: center;
  color: #fff;
  letter-spacing: 1px;
  text-decoration: none;
  line-height: 23px;
  font-size: 10px;
  display: block;
  margin: 30px;
  text-shadow: -1px -1px 0 #a84155;
  background: #d25068;
  border: 1px solid #d25068;
  width: 12rem;
  background-image: linear-gradient(to bottom, #f66c7b, #d25068);
  border-radius: 5px;
  box-shadow: 0 1px 0 rgba(255, 255, 255, 0.5) inset,
    0 -1px 0 rgba(255, 255, 255, 0.1) inset, 0 4px 0 #ad4257,
    0 4px 2px rgba(0, 0, 0, 0.5);
}

.button:before {
  background: #f0f0f0;
  background-image: linear-gradient(#d0d0d0, #f0f0f0);
  border-radius: 5px;
  box-shadow: 0 1px 2px rgba(0, 0, 0, 0.5) inset, 0 1px 0 #fff;
  position: absolute;
  content: "";
  left: -6px;
  right: -6px;
  top: -6px;
  bottom: -10px;
  z-index: -1;
}

.button:active {
  box-shadow: 0 1px 0 rgba(255, 255, 255, 0.5) inset,
    0 -1px 0 rgba(255, 255, 255, 0.1) inset;
  top: 5px;
}

.button:active:before {
  top: -11px;
  bottom: -5px;
  content: "";
}

.button:hover {
  background: #f66c7b;
  background-image: linear-gradient(top, #d25068, #f66c7b);
}

.end {
  background-image: url(/Users/limit/development/Tetris/images/blue_block.png);
}

.display {
  display: flex;
```

```css
  flex-direction: column;
  justify-content: space-between;
  align-items: center;
  text-align: center;
  margin-top: 1rem;
  width: 17.5rem;
  height: 25rem;
  background: #f0f0f0;
  background-image: linear-gradient(#d0d0d0, #f0f0f0);
  border-radius: 5px;
  box-shadow: 0 1px 2px rgba(0, 0, 0, 0.5) inset, 0 1px 0 #fff;
  color: #85796b;
}

.score,
.lines-display {
  padding-top: 1rem;
  font-size: 1.2rem;
}

/*menu*/
.container {
  max-width: 600px;
  padding: 0 3rem;
  margin: auto;
  overflow: hidden;
}

.btn:hover {
  opacity: 0.7;
}


.menu-wrap {
  position: fixed;
  top: 0;
  left: 0;
  z-index: 1;
}
```

```
.menu-wrap .toggler {
  position: absolute;
  top: 0;
  left: 0;
  z-index: 2;
  width: 50px;
  height: 50px;
  opacity: 0;
  cursor: pointer;
}

.menu-wrap .hamburger {
  position: absolute;
  top: 0;
  left: 0;
  z-index: 1;
  display: flex;
  width: 40px;
  height: 40px;
  padding: 1rem;
  background: rgba(13, 110, 139, 0.75);
  align-items: center;
  justify-content: center;
}


.menu-wrap .hamburger > div {
  position: relative;
  display: flex;
  width: 150%;
  height: 2px;
  background: #fff;
  flex: none;
  align-items: center;
  justify-content: center;
  transition: all 0.4s ease;
}


.menu-wrap .hamburger > div:before,
```

```css
.menu-wrap .hamburger > div:after {
  position: absolute;
  top: -7px;
  z-index: 1;
  width: 100%;
  height: 2px;
  background: inherit;
  content: "";
}


.menu-wrap .hamburger > div:after {
  top: 7px;
}


.menu-wrap .toggler:checked + .hamburger > div {
  transform: rotate(135deg);
}


.menu-wrap .toggler:checked + .hamburger > div:before,
.menu-wrap .toggler:checked + .hamburger > div:after {
  top: 0;
  transform: rotate(90deg);
}


.menu-wrap .toggler:checked:hover + .hamburger > div {
  transform: rotate(225deg);
}

.menu {
  display: flex;
  justify-content: center;
  position: fixed;
  z-index: 1;
  left: 0;
  top: 0;
  width: 100%;
  height: 100%;
  overflow: auto;
  background-color: rgba(24, 39, 51, 0.85);
}

.menu-content {
  text-align: center;
  width: 600px;
  align-items: center;
  margin-top: 230px;
  justify-content: center;
  width: 200vw;
  height: 200vh;
  border-radius: 50%;
  transition: all 0.8s ease;
}

.rules {
  font-size: 12px;
  transition: color 0.4s ease;
}

.key {
  color: #f8de7e;
}

.close {
  border-radius: 5px;
  color: rgba(24, 39, 51, 0.85);
}
```

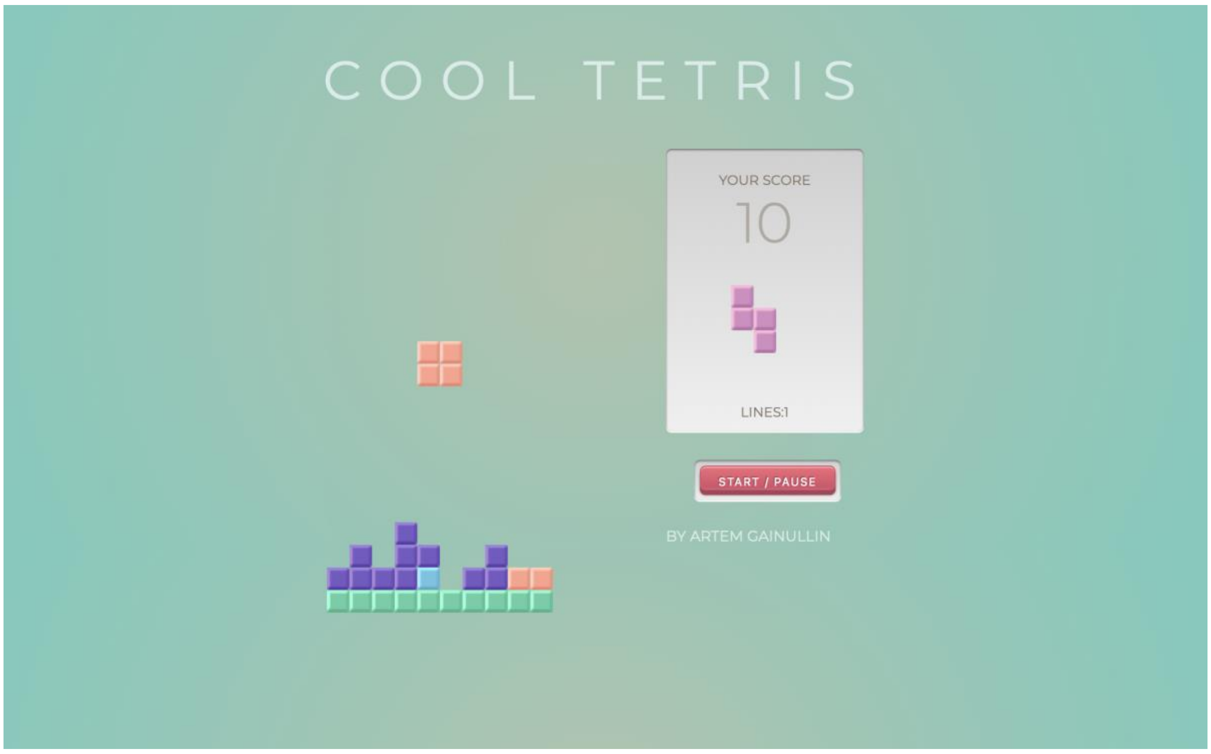
Файл app.js

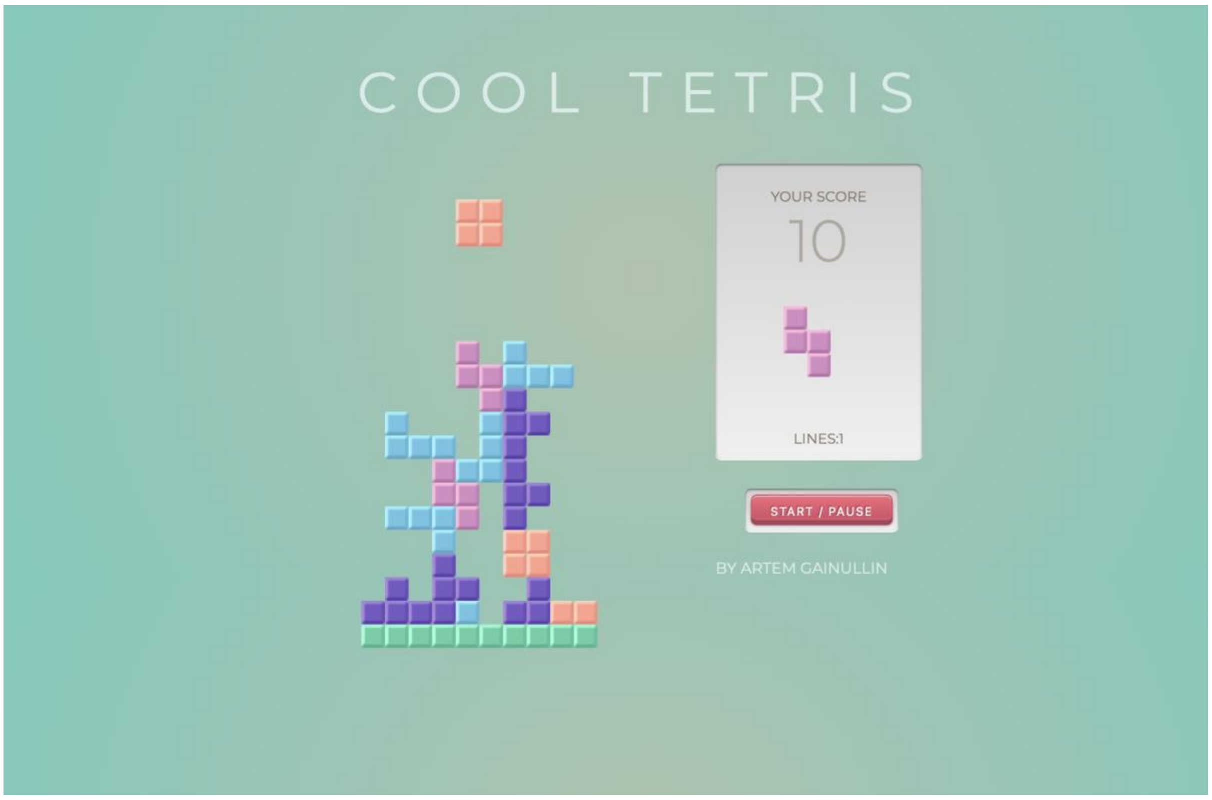
```javascript
document.addEventListener("DOMContentLoaded", () => {
  const GRID_WIDTH = 10;
  const GRID_HEIGHT = 20;
  const GRID_SIZE = GRID_WIDTH * GRID_HEIGHT;

  const grid = createGrid();
  let squares = Array.from(grid.querySelectorAll("div"));
  const startBtn = document.querySelector(".button");
  const hamburgerBtn = document.querySelector(".toggler");
  const menu = document.querySelector(".menu");
  const span = document.getElementsByClassName("close")[0];
  const scoreDisplay = document.querySelector(".score-display");
  const linesDisplay = document.querySelector(".lines-score");
  let currentIndex = 0;
  let currentRotation = 0;
  const width = 10;
  let score = 0;
  let lines = 0;
  let timerId;
  let nextRandom = 0;
  const colors = [
    "url(images/blue_block.png)",
    "url(images/pink_block.png)",
    "url(images/purple_block.png)",
    "url(images/peach_block.png)",
    "url(images/yellow_block.png)",
  ];

  function createGrid() {
    // основное поле
    let grid = document.querySelector(".grid");
    for (let i = 0; i < GRID_SIZE; i++) {
      let gridElement = document.createElement("div");
      grid.appendChild(gridElement);
    }


    for (let i = 0; i < GRID_WIDTH; i++) {
      let gridElement = document.createElement("div");
      gridElement.setAttribute("class", "block3");
      grid.appendChild(gridElement);
    }

    let previousGrid = document.querySelector(".previous-grid");

    for (let i = 0; i < 16; i++) {
      let gridElement = document.createElement("div");
      previousGrid.appendChild(gridElement);
    }
    return grid;
   }

   //управление
   function control(e) {
    if (e.keyCode === 39) moveright();
    else if (e.keyCode === 38) rotate();
    else if (e.keyCode === 37) moveleft();
    else if (e.keyCode === 40) moveDown();
   }

   // ускорение при зажатии клавиши вниз
   document.addEventListener("keydown", control);
  //мои фигуры
   const lTetromino = [
    [1, GRID_WIDTH + 1, GRID_WIDTH * 2 + 1, 2],
    [GRID_WIDTH, GRID_WIDTH + 1, GRID_WIDTH + 2, GRID_WIDTH * 2 + 2],
    [1, GRID_WIDTH + 1, GRID_WIDTH * 2 + 1, GRID_WIDTH * 2],
    [GRID_WIDTH, GRID_WIDTH * 2, GRID_WIDTH * 2 + 1, GRID_WIDTH * 2 + 2],
   ];

   const zTetromino = [
    [0, GRID_WIDTH, GRID_WIDTH + 1, GRID_WIDTH * 2 + 1],
    [GRID_WIDTH + 1, GRID_WIDTH + 2, GRID_WIDTH * 2, GRID_WIDTH * 2 + 1],
    [0, GRID_WIDTH, GRID_WIDTH + 1, GRID_WIDTH * 2 + 1],
    [GRID_WIDTH + 1, GRID_WIDTH + 2, GRID_WIDTH * 2, GRID_WIDTH * 2 + 1],
   ];

   const tTetromino = [
    [1, GRID_WIDTH, GRID_WIDTH + 1, GRID_WIDTH + 2],
    [1, GRID_WIDTH + 1, GRID_WIDTH + 2, GRID_WIDTH * 2 + 1],
    [GRID_WIDTH, GRID_WIDTH + 1, GRID_WIDTH + 2, GRID_WIDTH * 2 + 1],
    [1, GRID_WIDTH, GRID_WIDTH + 1, GRID_WIDTH * 2 + 1],
   ];

   const oTetromino = [
    [0, 1, GRID_WIDTH, GRID_WIDTH + 1],
    [0, 1, GRID_WIDTH, GRID_WIDTH + 1],
    [0, 1, GRID_WIDTH, GRID_WIDTH + 1],
    [0, 1, GRID_WIDTH, GRID_WIDTH + 1],
  ];

  const iTetromino = [
    [1, GRID_WIDTH + 1, GRID_WIDTH * 2 + 1, GRID_WIDTH * 3 + 1],
    [GRID_WIDTH, GRID_WIDTH + 1, GRID_WIDTH + 2, GRID_WIDTH + 3],
    [1, GRID_WIDTH + 1, GRID_WIDTH * 2 + 1, GRID_WIDTH * 3 + 1],
    [GRID_WIDTH, GRID_WIDTH + 1, GRID_WIDTH + 2, GRID_WIDTH + 3],
  ];

  const theTetrominoes = [
    lTetromino,
    zTetromino,
    tTetromino,
    oTetromino,
    iTetromino,
  ];

  //Выбираем случайную фигуру
  let random = Math.floor(Math.random() * theTetrominoes.length);
  let current = theTetrominoes[random][currentRotation];

  //движение вниз
  let currentPosition = 4;
  //отрисовка
  function draw() {
    current.forEach((index) => {
      squares[currentPosition + index].classList.add("block");
      squares[currentPosition + index].style.backgroundImage = colors[random];
    });
  }


  function undraw() {
    current.forEach((index) => {
      squares[currentPosition + index].classList.remove("block");
      squares[currentPosition + index].style.backgroundImage = "none";
    });
  }
```

```
//постоянное движение фигур вниз
function moveDown() {
  undraw();
  currentPosition = currentPosition += width;
  draw();
  freeze();
}

startBtn.addEventListener("click", () => {
  if (timerId) {
    clearInterval(timerId);
    timerId = null;
  } else {
    draw();
    timerId = setInterval(moveDown, 1000);
    nextRandom = Math.floor(Math.random() * theTetrominoes.length);
    displayShape();
  }
});

//движение влево и ограничения
function moveright() {
  undraw();
  const isAtRightEdge = current.some(
    (index) => (currentPosition + index) % width === width - 1
  );
  if (!isAtRightEdge) currentPosition += 1;
  if (
    current.some((index) =>
      squares[currentPosition + index].classList.contains("block2")
    )
  ) {
    currentPosition -= 1;
  }
  draw();
}

//движение вправо и ограничения
function moveleft() {
  undraw();
  const isAtLeftEdge = current.some(
    (index) => (currentPosition + index) % width === 0
  );
  if (!isAtLeftEdge) currentPosition -= 1;
  if (
    current.some((index) =>
      squares[currentPosition + index].classList.contains("block2")
    )
  ) {
    currentPosition += 1;
  }
  draw();
}

//остановка
function freeze() {
  if (
    current.some(
      (index) =>
        squares[currentPosition + index + width].classList.contains(
          "block3"
        ) ||
        squares[currentPosition + index + width].classList.contains("block2")
    )
  ) {
    current.forEach((index) =>
      squares[index + currentPosition].classList.add("block2")
    );
    // запуск новой фигруы
    random = nextRandom;
    nextRandom = Math.floor(Math.random() * theTetrominoes.length);
    current = theTetrominoes[random][currentRotation];
    currentPosition = 4;
    draw();
    displayShape();
    addScore();
    gameOver();
  }
}
freeze();
```

```
//Поворот фигур
function rotate() {
  undraw();
  currentRotation++;
  if (currentRotation === current.length) {
    currentRotation = 0;
  }
  current = theTetrominoes[random][currentRotation];
  draw();
}

//Конец игры
function gameOver() {
  if (
    current.some((index) =>
      squares[currentPosition + index].classList.contains("block2")
    )
  ) {
    scoreDisplay.innerHTML = "end";
    clearInterval(timerId);
  }
}

//sПоказ фигруы на экране
const displayWidth = 4;
const displaySquares = document.querySelectorAll(".previous-grid div");
let displayIndex = 0;

const smallTetrominoes = [
  [1, displayWidth + 1, displayWidth * 2 + 1, 2] ,
  [0, displayWidth, displayWidth + 1, displayWidth * 2 + 1] ,
  [1, displayWidth, displayWidth + 1, displayWidth + 2] ,
  [0, 1, displayWidth, displayWidth + 1] ,
  [
    1,
    displayWidth + 1,
    displayWidth * 2 + 1,
    displayWidth * 3 + 1,
  ] ,
  ];
// дисплей со счетом и фигурой
  function displayShape() {
    displaySquares.forEach((square) => {
      square.classList.remove("block");
      square.style.backgroundImage = "none";
    });
    smallTetrominoes[nextRandom].forEach((index) => {
      displaySquares[displayIndex + index].classList.add("block");
      displaySquares[displayIndex + index].style.backgroundImage =
        colors[nextRandom];
    });
  }

  //добавление счёта
  function addScore() {
    for (
      currentIndex = 0;
      currentIndex < GRID_SIZE;
      currentIndex += GRID_WIDTH
    ) {
      const row = [
        currentIndex,
        currentIndex + 1,
        currentIndex + 2,
        currentIndex + 3,
        currentIndex + 4,
        currentIndex + 5,
        currentIndex + 6,
        currentIndex + 7,
        currentIndex + 8,
        currentIndex + 9,
      ];
      if (row.every((index) => squares[index].classList.contains("block2"))) {
        score += 10;
        lines += 1;
        scoreDisplay.innerHTML = score;
        linesDisplay.innerHTML = lines;
        row.forEach((index) => {
          squares[index].style.backgroundImage = "none";
          squares[index].classList.remove("block2") ||
            squares[index].classList.remove("block");
        });
        const squaresRemoved = squares.splice(currentIndex, width);
        squares = squaresRemoved.concat(squares);
        squares.forEach((cell) => grid.appendChild(cell));
      }
    }
  }

  hamburgerBtn.addEventListener("click", () => {
    menu.style.display = "flex";
  });
  span.addEventListener("click", () => {
    menu.style.display = "none";
  });
});
```

Пример выполнения программы




COOL TETRIS
YOUR SCORE
10
LINES:1
START / PAUSE
BY ARTEM GAINULLIN