УДК 004

Создание эффекта перемещения частиц на JavaScript

Стрельцова Марина Николаевна
Приамурский государственный университет им. Шолом-Алейхема
Студент

Аннотация
В современным мире простые Web-страниц не являются конкурентоспособными, сейчас пользователям интересно смотреть на более эффектные, яркие и красочные страницы. Для создания интерактивных web-страниц чаще всего применяется язык программирования JavaScript. В данной статье разработан эффект перемещения частиц на языке программирования JavaScript.
Ключевые слова: эффект перемещения, JavaScript , Html, CSS

Creating a particle movement effect in JavaScript

Streltsova Marina Nikolaevna
Sholom-Aleichem Priamursky State University
Student

Abstract
In the modern world, simple Web pages are not competitive; now users are interested in looking at more spectacular, vibrant and colorful pages. The JavaScript programming language is most often used to create interactive web pages. This article has developed the effect of moving particles in the JavaScript programming language.
Keywords: move effect, JavaScript, Html, CSS

В современном информационном обществе специалист в любой сфере деятельности должен уметь создавать Web-страницы, которые являются основными ресурсами глобальной компьютерной сети Интернет и широко используются во всемирной паутине. Одним из основных средств создания Web-страниц является язык разметки гипертекста HTML. Однако, для создания интерактивных Web-приложений этого недостаточно. Современными средствами для создания интерактивных Web-приложений являются языки программирования JavaScript и PHP. В данной статье будет описана разработка эффекта перемещения частиц на языке программирования JavaScript.

В статье В.В. Кургасова и А.Г. Лапшовой рассмотрены наиболее известные JavaScript фреймворки на 2018 год [1]. Научная работа А.А. Леванова и А. Ю. Винокурова посвещена созданию универсальной программной платформы для разработки мультимедийных электронных

учебников на базе Javascript и HTML5 [2]. D. H. Mosley и J. M. André описывают использование JavaScript в примерах квантово-химических приложений, предназначенных для образовательных целей [3]. Создание игрового движка на языках программирования HTML5 и JavaScript описано в статье Д. Д. Трескина [4]. В исследовании А.Е. Болгова рассматривается использование спрайтов для создания анимации на веб-страницах, так же описываются преимущества данного метода на примере демонстрации техники анимации с использованием спрайт-листа [5]. E. Fritz и T. Zhao в своей статье представляют семантику стрелочного DSL в JavaScript, который может кодировать асинхронные программы как конечный автомат, где краевые переходы запускаются внешними событиями [6]. В статье Н.С. Колышкиной и Л.Е. Малковой рассматриваются особенности создания анимированного фавикона для web-ресурсов [7].

Целью данной статья является создание эффекта перемещения частиц на языке программирования JavaScript.

JavaScript (JS) — это полноценный динамический язык программирования, который применяется к HTML документу и добавляет интерактивность на веб-сайт [8].

В начале работы создаем три файла: index.html, style.css, script.js.

Теперь файл index.html заполняем базовой структурой документа и добавляем строки подключения css стиля и скрипта js (Рис.1).

```
<!DOCTYPE html>
<html lang="ru">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Effect 1</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <canvas></canvas>
    <script src="script.js"></script>
</body>
</html>
```

Рисунок 1 – Код документа index.html

Далее добавим несколько строк кода в style.css (Рис. 2)

```
1  body {
2      margin:          0%;
3      overflow:        hidden;
4  }
5  canvas {
6      background: rgb(0, 0, 0);
7  }
```

Рисунок 2 – Код документа style.css

Теперь опишем структуру js файла, который создаст нужную анимацию. В начале напишем самовызывающуюся функцию, где получим данные canvas и запишем высоту и ширину в отдельные переменные. Также занесем в переменные центр по x и y и вызовем функцию. Цвет фона будет серым, который будет растягиваться по всей ширине окна при его уменьшении или увеличении (Рис. 3).

```
(() => {

  const cnv  = document.querySelector('canvas');
  const ctx  = cnv.getContext('2d');

  let cw, ch, cx, cy;
  function resizeCanvas() {
    cw = cnv.width  = innerWidth ;
    ch = cnv.height = innerHeight;
    cx = cw / 2;
    cy = ch / 2;
  }
  resizeCanvas();
  window.addEventListener(`resize`, resizeCanvas);
```

Рисунок 3 – Самовызывающаяся функция

Создадим класс Dot, который будет отвечать за частицу, задавая ее начальное положение (Рис. 4).

```
  class Dot {
    constructor() {
      this.pos     = {x: cx, y: cy};
      this.dir     = (Math.random() * 3 | 0) * 2;
      this.step    = 0;
    }
```

Рисунок 4 – Класс Dot

Напишем функцию drawRect для отрисовки частицы (Рис. 5).

```
function drawRect(color, x, y, w, h, shadowColor, shadowBlur, gco) {
  ctx.globalCompositeOperation = gco;
  ctx.shadowColor = shadowColor || `black`;
  ctx.shadowBlur  = shadowBlur  || 1;
  ctx.fillStyle   = color;
  ctx.fillRect(x, y, w, h);
}
```

Рисунок 5 – Функция drawRect

Следующим шагом будет написание функции для отображения частицы в браузере (Рис. 6).

```
43    redrawDot() {
44      let xy       = Math.abs(this.pos.x - cx) + Math.abs(this.pos.y - cy);
45      let makeHue  = (cfg.hue + xy / cfg.gradientLen ) % 360;
46      let color    = `hsl(${ makeHue }, 100%, 50%)`;
47      let blur     = cfg.dotSize - Math.sin(xy / 8) * 2;
48      let size     = cfg.dotSize;// - Math.sin(xy / 9) * 2 + Math.sin(xy / 2);
49      let x        = this.pos.x - size / 2;
50      let y        = this.pos.y - size / 2;
51
52      drawRect(color, x, y, size, size, color, blur, `lighter`);
53    }
```

Рисунок 6 – Функция redrawDot

Для изменения положения частицы по осям x или y напишем функции MoveDot и loop (Рис. 7).

```
55      moveDot() {
56        this.step++;
57        this.pos.x += dirsList[this.dir].x * cfg.dotVelocity;
58        this.pos.y += dirsList[this.dir].y * cfg.dotVelocity;
59      }
60      function loop() {
61      drawRect(cfg.bgFillColor, 0, 0, cw, ch, 0, 0, `normal`);
62      addDot();
63      refreshDots();
64
65      requestAnimationFrame(loop);
66    }
67    loop();
```

Рисунок 7 – Функции moveDot и loop

Вынесем в отдельную константу все основные настройки частиц: цвет, количество, скорость и другие (Рис. 8).

```
16    const cfg = {
17      hue          : 0,
18      bgFillColor  : `rgba(50, 50, 50, .01)`,
19      dirsCount    : 6,
20      stepToTurn   : 8,
21      dotSize      : 2,
22      dotsCount    : 300,
23      dotVelocity  : 2,
24      distance     : 70,
25      gradientLen  : 5,
26    }
```

Рисунок 8 – Настройки частиц

Далее опишем функцию createDirs, которая будет отвечать за начальное направление движения частицы по сетке. Полученные значения x и y занесем в массив dirsList (Рис. 9).

```
let dirsList = [];
function createDirs() {
  for (let i = 0 ; i < 360 ; i+= 360 / cfg.dirsCount) {
    let x = Math.cos(i * Math.PI / 180);
    let y = Math.sin(i * Math.PI / 180);
    dirsList.push({x: x, y: y});
  }
}
createDirs();
```

Рисунок 9 – Функция dirsList

Функция changeDir будет менять направление движения частицы (Рис. 10).

```
changeDir() {
  if (this.step % cfg.stepToTurn === 0) {
    this.dir     = Math.random() > 0.5 ? (this.dir + 1) % cfg.dirsCount : (this.dir + cfg.dirsCount - 1)
  }
}
```

Рисунок 10 – Функция changeDir

Функция addDot добавляет частицы с некоторой вероятностью (Рис. 11)

```
let dotsList = [];
function addDot() {
  if (dotsList.length < cfg.dotsCount && Math.random() > .8) {
    dotsList.push( new Dot() );
    cfg.hue = (cfg.hue + 1) % 360 ;
  }
}
```

Рисунок 11 – Функция addDot

Далее напишем функцию противоположную addDot, отвечающую за уничтожение частиц через некоторый период с конкретным id (Рис. 12).

```
killDot(id) {
  let percent = Math.random() * Math.exp(this.step / cfg.distance);
  if (percent > 100) {
    dotsList.splice(id, 1);
  }
}
}
```

Рисунок 12 – Функция killDot

Функция refreshDots служит для присвоения частице ранее добавленных методов (Рис. 13).

```
function refreshDots() {
  dotsList.forEach( (i, id) => {
    i.redrawDot();
    i.moveDot();
    i.changeDir();
    i.killDot(id)
  } );
}
```

Рисунок 13 – Функция refreshDots

Запустим проект и посмотрим на анимацию (Рис. 14).

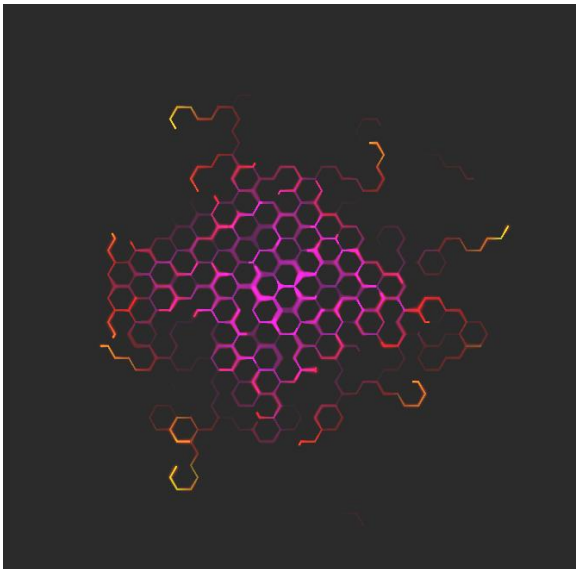

Рисунок 14 – Полученная анимация

Также изменяя параметры константы cfg можно получить анимации различного вида (Рис. 15-16).

```
const cfg = {
  hue          : 0,
  bgFillColor  : `rgba(50, 50, 50, .01)`,
  dirsCount    : 3,
  stepToTurn   : 20,
  dotSize      : 2,
  dotsCount    : 300,
  dotVelocity  : 2,
  distance     : 200,
  gradientLen  : 5,
}
```

Рисунок 15 – Измененная константа cfg

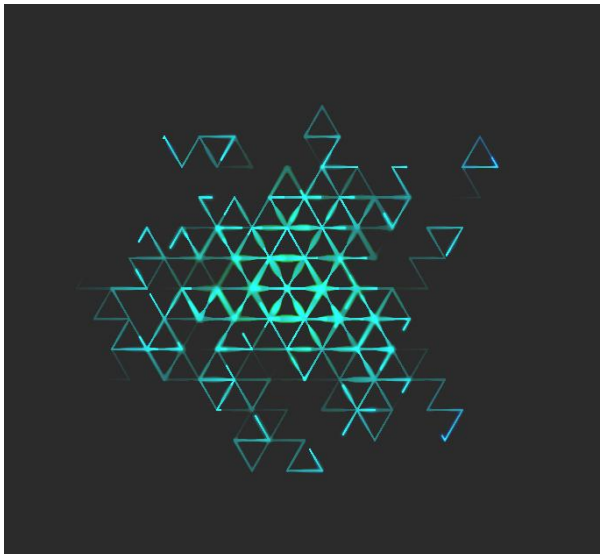

Рисунок 16 – Полученная анимация при изменении константы cfg

Таким образом в рамках данного исследования был создан эффект перемещения частиц с помощью JavaScript, а также рассмотрены особенности реализации движения частиц в различных направлениях.

Библиографический список

1. Кургасов В. В., Лапшова А. Г. JavaScript фреймворки // Центральный научный вестник. 2018. Т. 3. №. 15-16. С. 40-41.
2. Леванов А. А., Винокуров А. Ю. Платформа для разработки мультимедийных электронных учебников на базе Javascript и HTML5 // Электронное обучение в непрерывном образовании. 2016. №. 1. С. 279-286.
3. Mosley D. H., André J. M. Use of JavaScript in simple quantum-chemical applications // Journal of Molecular Structure: THEOCHEM. 1997. Т. 419. №. 1-3. С. 57-62.
4. Трескин Д. Д. Создание игрового движка средствами html5 и javascript // Интеллектуальный потенциал XXI века инновационной России. 2019. С. 74-77.
5. Болгов А.Е. Создание покадровой анимации с помощью спрайтов // Научно-практические исследования. 2020. №8-8 (31) . С. 20-22.
6. Fritz E., Zhao T. Typing and semantics of asynchronous arrows in JavaScript // Science of Computer Programming. 2017. Т. 141. С. 1-39.
7. Колышкина Н.С., Малкова Л.Е. Вариант разработки анимированного фавикона // Modern Science. 2020. №6-2. с. 263-267.
8. JavaScript.com URL: https://www.javascript.com/ (дата обращения: 19.01.2021).