Информационно-коммуникационные технологии в педагогическом образовании, 2025. № 1 (94). *infed.ru*

УДК 378.016:004.42

С. В. Журавлев

S. V. Zhuravlev

Журавлев Сергей Владимирович, преподаватель, КГПИ ФГБОУ ВО «КемГУ», г. Новокузнецк, Россия.

Zhuravlev Sergei Vladimirovich, Lecturer, Kuzbass Humanitarian Pedagogical Institute of Kemerovo State University, Novokuznetsk, Russia.

МЕТОДИКА ИЗУЧЕНИЯ ТЕМЫ «ГРАФИКА И ПОДПРОГРАММЫ» В КУРСЕ ДИСЦИПЛИНЫ «ПРОГРАММИРОВАНИЕ НА JAVASCRIPT»

METHODOLOGY FOR STUDYING THE TOPIC «GRAPHICS AND SUBPROGRAMS» IN THE COURSE OF THE DISCIPLINE «JAVASCRIPT PROGRAMMING»

Аннотация. *В статье представлена авторская методика обучения студентов направленности «Компьютерный дизайн» дисциплине «Программирование на JavaScript», которая подразумевает подробное объяснение теоретического материала с разбором решения некоторых задач. Приводится пример изучения темы « Графика и подпрограммы».*

Annotation. *The article presents the author's methodology for teaching students of the «Computer Design» focus the discipline «JavaScript programming», which involves a detailed explanation of the theoretical material with an analysis of some problem solving. An example of studying the topic «Graphics and subprograms» is given.*

Ключевые слова: *обучение скриптовому программированию, язык программирования JavaScript, методы решения задач, холст, программирование графики.*

Keywords: *script programming training, JavaScript programming language, methods of solving problems, canvas, graphics programming.*

В профессиональную подготовку бакалавров по направлению 44.03.04 «Профессиональное обучение (по отраслям)», направленности «Компьютерный дизайн», входит ряд дисциплин, способствующих формированию профессиональной компетенции ПК-1 «Способен осваивать и использовать теоретические знания и практические умения и навыки в предметной области по профилю «Компьютерный дизайн» при решении профессиональных задач», в том числе дисциплина вариативной части блока Б1 К.М.07.01.10 «Программирование на JavaScript».

Целью данной дисциплины является формирование индикатора достижения компетенции ПК-1.3 «Демонстрирует методы использования программных и аппаратных средств для создания объектов компьютерного дизайна» [4]. Для успешного формирования этого индикатора преподавателю необходимо качественно выстраивать учебный процесс, излагать материал достаточно подробно и в доступной форме.

В предыдущей статье была рассмотрена методика изучения темы «Одномерные массивы» [1]. В данной статье речь пойдёт о методике изучения темы «Графика и подпрограммы» в курсе дисциплины «Программирование на JavaScript». Изучение данной темы очень важно, т. к. студенты приобретают опыт графического оформления веб-страниц, что позволяет расширить владение ими средствами, которые могут быть использованы в профессиональной деятельности.

Изучение темы начинается с объяснения преподавателем теоретического материала, необходимого для работы с графикой.

В HTML5 есть тег **canvas (холст)**, который создаёт поле для рисования. На нём можно рисовать с помощью скриптов JavaScript.

Сначала необходимо на странице прописать внутри тега `<body></body>` тег:

```
<canvas id="canvas" width="1900" height="800" style = "border: solid"></canvas>
```

Следует отметить, что обязательным атрибутом тега является только идентификатор **id**, остальные можно как указывать, так и не указывать.

Атрибуты **width** и **height** задают соответственно ширину и высоту холста. Если размеры не указать, то по умолчанию размер холста будет равен 300x150 пикселей. Canvas создаёт область фиксированного размера, содержимым которого управляют контексты. Атрибуты не имеют отношения к CSS, они обозначают ширину и высоту canvas в пикселях не на экране, а на координатной плоскости холста.

Атрибут **style** позволяет применить стиль. В приведённом примере он создаёт сплошную рамку вокруг холста.

Теперь с помощью JavaScript надо обратиться к этому полю. Сначала нужно получить ссылку на него через **getElementById**, а затем создать переменную, в которой будет находиться так называемый «контекст», через методы которого и происходит рисование:

```
let canvas, im_1;

canvas = document.getElementById('canvas');

im_1 = canvas.getContext('2d');
```

Теперь, обращаясь к методам в переменной im_1, можно рисовать различные фигуры в пределах области canvas.

На экране **координату (0, 0)** имеет точка, расположенная в **левом верхнем углу** объекта для рисования, в данном случае canvas. Соответственно, **экранная абсцисса** (координата **x**) увеличивается **слева направо**, а **экранная ордината** (координата **y**) увеличивается **сверху вниз**.

Далее рассматриваются свойства и методы для рисования.

Свойства **canvas.width** и **canvas.height** вычисляют размер холста.

Все методы будут относиться к переменной **im_1**.

Метод **moveTo(x, y) –** перемещает «курсор» в позицию x, y и делает её текущей.

Метод **lineTo(x, y) –** ведёт линию из текущей позиции в указанную, и делает впоследствии указанную позицию текущей.

После метода moveTo можно сколько угодно раз использовать метод lineTo для рисования «цепочки» линий. Для того чтобы нарисовать замкнутую фигуру, можно вместо последнего использования метода lineTo применить метод **closePath()**, который рисует линию, автоматически вычисляя нужные координаты, чтобы сомкнуть её с началом рисунка.

Однако использования этих трёх методов недостаточно для того, чтобы линия отобразилась на экране. Для этого после этих методов нужно объявить об окончании рисования одним из следующих способов:

1) с помощью метода **stroke()**, который просто делает линии видимыми;

2) с помощью метода **fill()**, который заливает фигуру сплошным цветом. При этом необязательно, чтобы фигура была закончена и «закрыта». То есть можно не ставить метод closePath, а использовать вместо него метод fill, тогда всё равно закрасится именно нужная область.

Перед использованием метода moveTo желательно объявить начало рисования линии или фигуры с помощью метода **beginPath().** Если этот метод не использовать, то линии всё равно отобразятся, но результат может оказаться не таким, на который рассчитывали.

Метод **strokeRect(x, y, w, h) –** рисует контуры прямоугольника, где x, y – координаты верхнего левого угла; w, h – ширина и высота.

Метод **fillRect(x, y, w, h) –** рисует закрашенный прямоугольник.

Метод **clearRect(x, y, w, h) –** очищает область на холсте прямоугольником заданного размера в указанной позиции, т. е. как бы вырезает кусок фигуры, за которым можно увидеть холст.

Метод **ellipse(x, y, radiusX, radiusY, rotation, startAngle, endAngle [, anticlockwise])** – задаёт (но не рисует!) эллиптическую дугу, где: x, y – координаты центра эллипса; radiusX, radiusY – радиусы по осям X и Y соответственно; rotation – угол поворота эллипса (в радианах); startAngle, endAngle – начальный и конечный углы дуги эллипса (в радианах); необязательный параметр anticlockwise – направление рисования дуги – по часовой стрелке (false, по умолчанию) или против часовой стрелки (true).

Важно отметить, что параметры startAngle и endAngle начинаются с направления оси X вдоль положительного направления оси Y (т. е. углы увеличиваются по часовой стрелке).

Если после метода ellipse добавить метод closePath, то нарисуется линия, соединяющая начало и конец дуги. Если перед методом ellipse указать точку методом moveTo, то эта точка соединится с началом дуги. При совместном использовании методов moveTo и closePath точка соединится и с началом, и с концом дуги.

Видимой дуга станет, если применить к ней методы stroke (только контур) или fill (с заливкой). А перед использованием метода ellipse следует добавлять метод beginPath.

Если значение параметра startAngle будет равно **0**, а значение параметра endAngle – **2*Math.PI** или наоборот, то нарисуется эллипс.

Если параметры radiusX и radiusY задать одинаковыми, то получится дуга окружности, но рисовать её методом ellipse нецелесообразно, т. к. для этого есть специальный метод с меньшим числом параметров.

Метод **arc(x, y, radius, startAngle, endAngle [, anticlockwise])** – задаёт дугу окружности, где: x, y – координаты центра окружности; radius – радиус; startAngle, endAngle – начальный и конечный углы дуги окружности (в радианах); необязательный параметр anticlockwise – направление рисования дуги – по часовой стрелке (false, по умолчанию) или против часовой стрелки (true).

Вся вышеперечисленная информация по методу ellipse относится и к методу arc.

По умолчанию для линий и заливки используется чёрный цвет.

Свойство **strokeStyle** позволяет изменить цвет линий, а свойство **fillStyle** – цвет заливки.

Существует 4 способа задания цвета:

1) непосредственно указать нужный цвет на английском языке, например: "orange";

2) в HEX формате, например: "#FFA500";

3) в обычном RGB формате, например: "rgb(255, 165, 0)";

4) в расширенном RGB формате, например: "rgba(255, 165, 0, 1)" – здесь последний параметр позволяет задавать полупрозрачность, принимает дробные значения от 0 до 1 (единица соответствует полной непрозрачности, а ноль – полной прозрачности) [5].

Затем начинается разбор типовых задач на графику с применением подпрограмм (внимание уделяется не всему коду программы, а только его фрагменту, непосредственно содержащему алгоритм решения конкретной задачи).

1. Нарисовать треугольник голубыми линиями.

```
im_1.strokeStyle = "blue";

im_1.beginPath();

im_1.moveTo(100, 300);

im_1.lineTo(200, 200);

im_1.lineTo(300, 300);

im_1.closePath();

im_1.stroke();
```

2. Нарисовать треугольник, закрашенный красным цветом.

```
im_1.fillStyle="red";

im_1.beginPath();

im_1.moveTo(100, 300);

im_1.lineTo(200, 200);
```

```
im_1.lineTo(300, 300);

im_1.fill();
```

3. Нарисовать прямоугольник жёлтыми линиями.

```
im_1.strokeStyle="yellow";

im_1.strokeRect(100, 350, 120, 100);
```

4. Нарисовать квадрат, закрашенный оранжевым цветом.

```
im_1.fillStyle="orange";

im_1.fillRect(100, 300, 150, 150);
```

5. Нарисовать эллипс зелёного цвета, повёрнутый на 45° против часовой стрелки.

```
im_1.strokeStyle="green";

im_1.beginPath();

im_1.ellipse(300, 600, 80, 150, Math.PI/4, 0, 2*Math.PI);

im_1.stroke();
```

6. Нарисовать круг, закрашенный голубым цветом.

```
im_1.fillStyle="blue";

im_1.beginPath();

im_1.arc(150, 50, 40, 0, 2*Math.PI);

im_1.fill();
```

7. Нарисовать дугу, представляющую собой четверть окружности от 0° до 90°, считая от верхней точки, и хорду красного цвета.

```
im_1.strokeStyle="red";

im_1.beginPath();

im_1.arc(150, 50, 40, 0, 3*Math.PI/2, true);
```

```
im_1.closePath();

im_1.stroke();
```

8. Нарисовать дугу, представляющую собой три четверти эллипса от 90° до 360°, считая от верхней точки, закрашенную жёлтым цветом.

```
im_1.fillStyle="yellow";

im_1.beginPath();

im_1.ellipse(300, 600, 80, 150, 0, 0, 3*Math.PI/2, false);

im_1.fill();
```

Затем подробно разбираются более сложные задачи.

1. Дана дуга, представляющая собой нижнюю полуокружность диаметром *d*, опирающуюся на диаметр, с контуром зелёного цвета. Расположить эти объекты на экране так, чтобы их центры находились в точках, разбивающих синусоиду, заданную на отрезке [-*π*, 0], и находящихся на одинаковом расстоянии друг от друга.

Прежде всего, следует обратить внимание студентов на то, что в задачах такого типа рисовать объект удобнее всего во вспомогательной функции, чтобы в основной программе при расположении множества таких объектов заданным образом не приходилось задавать координаты объекта длинными формулами. Поскольку подпрограммы ранее уже изучались в курсе дисциплине, это не должно составить особого труда.

Чтобы расположить объекты по синусоиде, нужно, по сути, построить невидимый график функции y = sin x, только вместо точек будут нарисованные объекты. Для построения графика любой функции необходимо использовать формулы перехода от декартовых координат к экранным:

$$\begin{cases} x_э = x_0 + x_д \cdot m \\ y_э = y_0 - y_д \cdot m \end{cases}$$, где:

$x_э$, $y_э$ – экранные координаты; $x_д$, $y_д$ – декартовы координаты; x_0, y_0 – координаты исходной точки графика на экране; m – коэффициент масштабирования, он нужен, чтобы график получился видимый глазу.

Во второй формуле стоит знак «-», потому что экранная ось ординат направлена в сторону, противоположную направлению декартовой оси ординат.

Решение

```
<!DOCTYPE HTML>
<html>
<head>
 <title>Проверка кода</title>
</head>
<body>
<canvas width="1900" height="800" id="canvas" style = "border: solid"></c
<script>
//создаём функцию для рисования нижней полуокружности с диам
зелёного цвета
function ris(x, y) {
im_1.beginPath();
im_1.strokeStyle="green";
im_1.arc(x, y, d/2, 0, Math.PI, false);
im_1.closePath();
im_1.stroke();
}
let canvas, im_1, n, d, m, x0, y0, x, i;
canvas=document.getElementById('canvas');
im_1=canvas.getContext('2d');
//очищаем всё пространство canvas, чтобы при перезапуске прог
предыдущий рисунок стирался
im_1.clearRect(0, 0, canvas.width, canvas.height);
//задаём количество объектов на синусоиде
n=parseInt(prompt("Введите количество объектов"));
d=parseFloat(prompt("Введите диаметр полуокружности"));
m=50;
//задаём значения координат исходной точки графика на экране – при
координаты центра canvas
x0=canvas.width/2-100;
y0=canvas.height/2;
//располагаем n объектов по синусоиде на одинаковом расстоянии д
друга
x=-Math.PI;
for (i=0; i<n; i++) {
ris(x0+x*m, y0-Math.sin(x)*m);
x+=Math.PI/(n-1);
```

2. Дан овал с контуром коричневого цвета с горизонтальным диаметром *a* и вертикальным диаметром *b (a<b)*. Расположить эти объекты на экране так, чтобы их центры находились в точках, разбивающих окружность радиуса *r*, и находящихся на одинаковом расстоянии друг от друга.

При расположении объектов по окружности формулы для вычисления экранных координат будут отличаться от соответствующих формул при расположении объектов по синусоиде или косинусоиде. В данном случае нужно использовать параметрическое уравнение окружности:

$$\begin{cases} x_э = x_0 + r \cdot \sin t \\ y_э = y_0 - r \cdot \cos t \end{cases}$$, где:

$x_Э$, $y_Э$ – экранные координаты точки на окружности; x_0, y_0 – экранные координаты центра окружности; r – радиус окружности; t – радианная мера дуги от верхней точки окружности до искомой точки по часовой стрелке.

Решение

```
<!DOCTYPE HTML>
<html>
<head>
 <title>Проверка кода</title>
</head>
<body>
<canvas width="1900" height="800" id="canvas" style = "border: solid"></can
<script>
//создаём функцию для рисования овала с контуром коричневого цвета
function ris(x, y) {
im_1.beginPath();
im_1.strokeStyle="brown";
im_1.ellipse(x, y, a/2, b/2, 0, 0, 2*Math.PI);
im_1.stroke();
}
let canvas, im_1, n, a, b, x0, y0, r, t, i;
canvas=document.getElementById('canvas');
im_1=canvas.getContext('2d');
//очищаем всё пространство canvas
im_1.clearRect(0, 0, canvas.width, canvas.height);
//задаём количество объектов на окружности
n=parseInt(prompt("Введите количество объектов"));
a=parseFloat(prompt("Введите горизонтальный диаметр овала a"));
b=parseFloat(prompt("Введите вертикальный диаметр овала b (b>a)"));
r=parseFloat(prompt("Введите радиус окружности"));
//задаём значения координат центра окружности
x0=canvas.width/2;
y0=canvas.height/2;
//располагаем n объектов по окружности на одинаковом расстоянии дру
друга
t=0;
for (i=0; i<n; i++) {
ris(x0+r*Math.sin(t), y0-r*Math.cos(t));
t+=2*Math.PI/n;
}
</script>
</body>
</html>
```

3. Дан правильный пятиугольник, заполненный красным цветом, вписанный в невидимую окружность радиуса *r*. Вывести на экран данные объекты так, чтобы их центры располагались в точках, лежащих на диагонали экрана, соединяющей его левую верхнюю и правую нижнюю точки, и находились на одинаковом расстоянии друг от друга.

В данной задаче вспомогательная окружность даётся для облегчения построения пятиугольника, потому что правильный (равносторонний) многоугольник (кроме квадрата) удобнее всего строить, представив его вписанным в окружность, разбитую настолько равных частей, сколько вершин у многоугольника. Таким образом, надо просто вычислить координаты вершин многоугольника с помощью параметрического уравнения окружности (см. задачу № 2) и соединить их.

Решение

```
<!DOCTYPE HTML>
<html>
<head>
 <title>Проверка кода</title>
</head>
<body>
<canvas width="1900" height="800" id="canvas" style = "border: solid"><
<script>
//создаём функцию для рисования правильного пятиугольника, запол
красным цветом
function ris(x, y) {
let t;
t=0;
im_1.fillStyle="red";
im_1.beginPath();
im_1.moveTo(x+r*Math.sin(t), y-r*Math.cos(t));
while (t<2*Math.PI) {
t+=2*Math.PI/5;
im_1.lineTo(x+r*Math.sin(t), y-r*Math.cos(t));
}
im_1.fill();
}
let canvas, im_1, n, x, y, r, i;
canvas=document.getElementById('canvas');
im_1=canvas.getContext('2d');
//очищаем всё пространство canvas
im_1.clearRect(0, 0, canvas.width, canvas.height);
//задаём количество объектов на диагонали
n=parseInt(prompt("Введите количество объектов"));
r=parseFloat(prompt("Введите радиус вспомогательной окружности"))
//задаём значения координат центра первого объекта
x=(canvas.width/n)/2;
y=(canvas.height/n)/2;
//располагаем n объектов по диагонали на одинаковом расстоянии
друга
for (i=0; i<n; i++) {
```

Далее студентам предлагается решить задачи самостоятельно. Приведём некоторые из них.

1. Дан квадрат, заполненный голубым цветом, со стороной *a*. Расположить эти объекты на экране так, чтобы их центры находились в точках, разбивающих косинусоиду, заданную на отрезке [0, 2*π*], и находящихся на одинаковом расстоянии друг от друга.

2. Дан овал, заполненный зелёным цветом, с горизонтальным диаметром *a* и вертикальным диаметром *b (a>b)*. Вывести на экран данные объекты так, чтобы их центры располагались в точках, лежащих на диагонали экрана, соединяющей его левую нижнюю и правую верхнюю точки, и находились на одинаковом расстоянии друг от друга.

3. Дан правильный треугольник с контуром зелёного цвета, вписанный в невидимую окружность радиуса *r1*. Расположить эти объекты на экране так, чтобы их центры находились в точках, разбивающих окружность радиуса *r2*, и находящихся на одинаковом расстоянии друг от друга.

4.* Написать программу, выводящую на экран рекурсивное построение.

В основании фигуры – квадрат (рис. 1) [3].

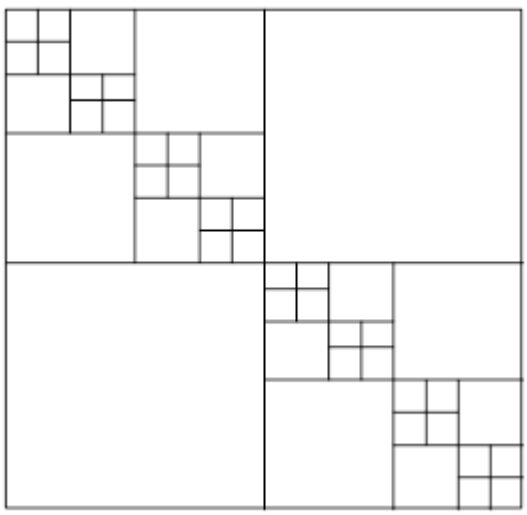

Рисунок 1. Изображение для задачи № 4

Предложенная методика изучения раздела «Графика» наряду с методикой изучения раздела «Массивы» является базой для дальнейшего изучения скриптового программирования, от полученных студентами компетенций зависит не только возможность дальнейшего изучения программирования, требующего освоить работу с графическими средствами JavaScript, но и овладение ими методикой обучения студентов СПО созданию объектов компьютерного дизайна при помощи JavaScript.

Кроме того, данная методика должна повысить интерес студентов к изучению программирования, способствовать стремлению научиться создавать сложные проекты, имеющие практическое применение, что, несомненно, будет способствовать повышению уровня их конкурентоспособности на рынке труда по окончании ВУЗа.

В настоящее время представленные материалы внедряются в образовательный процесс на факультете информатики, математики и экономики.

Список литературы

1. Журавлёв, С. В. Методика изучения темы «Одномерные массивы» в курсе дисциплины «Программирование на JavaScript» / С. В. Журавлёв. – Текст : электронный // Информационно-коммуникационные технологии в педагогическом образовании: электронный научный журнал, 2024. – № 01(88) январь. – URL : http://infed.ru/articles/10265/ (дата обращения : 17.02.2025).
2. Кингсли, Х. Э. JavaScript в примерах: учебное пособие / Х. Э. Кингсли, Х. К. Кингсли. – М.: ДМК Пресс, 2009. – 272 с. – ISBN 978-5-94074-668-3. – Текст : электронный // Лань: электронно-библиотечная система. – URL : https://e.lanbook.com/book/1271 (дата обращения : 11.12.2024). – Режим доступа: для авториз. пользователей.
3. Можаров, М. С. Введение в структурное программирование: Учебное пособие / М. С. Можаров, Г. Н. Бойченко. – 2-е изд., стереот. – Новокузнецк: Изд-во КузГПА, 2014. – 203 с. – Текст: непосредственный.

4. Основная профессиональная образовательная программа высшего образования: направление подготовки 44.03.04 «Профессиональное обучение (по отраслям)», направленность (профиль) «Компьютерный дизайн», уровень бакалавриата (утверждена 12 апреля 2023 г.). – Текст : электронный. – URL : https://skado.dissw.ru/edprogramdescriptionfile/2786/ (дата обращения : 11.12.2024).
5. Рисование в JavaScript (canvas). // Mousedc : [сайт]. – URL : https://www.mousedc.ru/learning/121-risovanie-javascript-canvas/ (дата обращения : 27.09.2024). – Текст : электронный.