Практическая работа

Курс: Разработка интерфейса на JavaScript

Дисциплина: Основы JavaScript

Тема занятия №23: Создание игры на JavaScript

Реализовать игру Змейка.

Логика игры

У классической змейки правила простые:

- есть поле из клеточек, где случайным образом появляется еда;
- есть змейка, которая всё время двигается и которой мы можем управлять;
- если змейка на своём пути встречает еду — еда исчезает, появляется в новом месте, а сама змейка удлиняется на одну клеточку;
- если змейка врежется в стену или в саму себя, игра заканчивается.

Чтобы играть было проще, мы сделаем так, чтобы змейка не врезалась в стенки, а проходила сквозь них. Если что — сможете это сами потом настроить в коде, когда захотите посложнее.

Последовательность наших действий будет такой:

1. Делаем пустую HTML-страницу.
2. Настраиваем внешний вид с помощью CSS.
3. Рисуем игровое поле.
4. Пишем скрипт, который и будет отвечать за всю игру.

Делаем HTML-страницу

С этим всё просто: берём стандартный код и сохраняем его как файл `snake.html`.

```html
<!DOCTYPE html>
<html>

<head>
  <title>Змейка</title>
  <style>
  </style>
</head>

<body>
  <!-- Содержимое страницы -->
</body>

</html>
```

Настраиваем внешний вид

За внешний вид на странице у нас отвечает раздел `<style>`, поэтому мы просто добавим в него CSS-код:

На сти
CSS-с

```
html,
body {
  height: 100%;
  margin: 0;
}
/*Задаём глобальные параметры*/
body {
  background: black;
  display: flex;
  align-items: center;
  justify-content: center;
}
/*Делаем границу вокруг игрового поля*/
canvas {
  border: 1px solid white;
}
```

Теперь у нас на странице нет лишних отступов, зато всё по центру, есть чёрный фон и граница вокруг игрового поля. Самое время создать само игровое поле.

Рисуем игровое поле

Поле делается очень просто:

```
<canvas id="game" width="400" height="400"></canvas>
```

400 пикселей в ширину, столько же в высоту, название поля — game. Этого достаточно, чтобы браузер отобразил холст с такими размерами и позволил нам на нём рисовать.

Пишем скрипт

1. Зададим все переменные, которые нам понадобятся.

JavaScript Copy

```
// Поле, на котором всё будет происходить, — тоже как бы переменная
var canvas = document.getElementById('game');
// Классическая змейка — двухмерная, сделаем такую же
var context = canvas.getContext('2d');
// Размер одной клеточки на поле — 16 пикселей
var grid = 16;
// Служебная переменная, которая отвечает за скорость змейки
var count = 0;
// А вот и сама змейка
var snake = {
  // Начальные координаты
  x: 160,
  y: 160,
  // Скорость змейки — в каждом новом кадре змейка смещается по оси X или У. На ст
  dx: grid,
  dy: 0,
  // Тащим за собой хвост, который пока пустой
  cells: [],
  // Стартовая длина змейки — 4 клеточки
  maxCells: 4
};
// А это — еда. Представим, что это красные яблоки.
var apple = {
  // Начальные координаты яблока
  x: 320,
  y: 320
};
```

2. Сделаем генератор случайных чисел. Он нам понадобится, чтобы размещать еду на поле случайным образом.

```
// Делаем генератор случайных чисел в заданном диапазоне

function getRandomInt(min, max) {

  return Math.floor(Math.random() * (max - min)) + min;

}
```

3. Напишем основной игровой цикл, который будет работать бесконечно.

JavaScript Copy

```
// Игровой цикл — основной процесс, внутри которого будет всё происходить
function loop() {
  // Дальше будет хитрая функция, которая замедляет скорость игры с 60 кадров в се
  requestAnimationFrame(loop);
  // Игровой код выполнится только один раз из четырёх, в этом и суть замедления к
  if (++count < 4) {
    return;
  }
  // Обнуляем переменную скорости
  count = 0;
  // Очищаем игровое поле
  context.clearRect(0, 0, canvas.width, canvas.height);
  // Двигаем змейку с нужной скоростью
  snake.x += snake.dx;
  snake.y += snake.dy;
  // Если змейка достигла края поля по горизонтали — продолжаем её движение с прот
  if (snake.x < 0) {
    snake.x = canvas.width - grid;
  }
  else if (snake.x >= canvas.width) {
    snake.x = 0;
  }
  // Делаем то же самое для движения по вертикали
  if (snake.y < 0) {
    snake.y = canvas.height - grid;
  }
  else if (snake.y >= canvas.height) {
    snake.y = 0;
  }
  // Продолжаем двигаться в выбранном направлении. Голова всегда впереди, поэтому
  snake.cells.unshift({ x: snake.x, y: snake.y });
  // Сразу после этого удаляем последний элемент из массива змейки, потому что она
  if (snake.cells.length > snake.maxCells) {
    snake.cells.pop();
  }
```

```
    // Рисуем еду — красное яблоко
    context.fillStyle = 'red';
    context.fillRect(apple.x, apple.y, grid - 1, grid - 1);
    // Одно движение змейки — один новый нарисованный квадратик
    context.fillStyle = 'green';
    // Обрабатываем каждый элемент змейки
    snake.cells.forEach(function (cell, index) {
      // Чтобы создать эффект клеточек, делаем зелёные квадратики меньше на один пик
      context.fillRect(cell.x, cell.y, grid - 1, grid - 1);
      // Если змейка добралась до яблока...
      if (cell.x === apple.x && cell.y === apple.y) {
        // увеличиваем длину змейки
        snake.maxCells++;
        // Рисуем новое яблочко
        // Помним, что размер холста у нас 400x400, при этом он разбит на ячейки — 2
        apple.x = getRandomInt(0, 25) * grid;
        apple.y = getRandomInt(0, 25) * grid;
      }
      // Проверяем, не столкнулась ли змея сама с собой
      // Для этого перебираем весь массив и смотрим, есть ли у нас в массиве змейки
      for (var i = index + 1; i < snake.cells.length; i++) {
        // Если такие клетки есть — начинаем игру заново
        if (cell.x === snake.cells[i].x && cell.y === snake.cells[i].y) {
          // Задаём стартовые параметры основным переменным
          snake.x = 160;
          snake.y = 160;
          snake.cells = [];
          snake.maxCells = 4;
          snake.dx = grid;
          snake.dy = 0;
          // Ставим яблочко в случайное место
          apple.x = getRandomInt(0, 25) * grid;
          apple.y = getRandomInt(0, 25) * grid;
        }
      }
    });
}
```

4. Сделаем управление стрелочками на клавиатуре.

```
// Смотрим, какие нажимаются клавиши, и реагируем на них нужным образом
document.addEventListener('keydown', function (e) {
  // Дополнительно проверяем такой момент: если змейка движется, например, влево,
  // Стрелка влево
  // Если нажата стрелка влево, и при этом змейка никуда не движется по горизонтал
  if (e.which === 37 && snake.dx === 0) {
    // то даём ей движение по горизонтали, влево, а вертикальное — останавливаем
    // Та же самая логика будет и в остальных кнопках
    snake.dx = -grid;
    snake.dy = 0;
  }
  // Стрелка вверх
  else if (e.which === 38 && snake.dy === 0) {
    snake.dy = -grid;
    snake.dx = 0;
  }
  // Стрелка вправо
  else if (e.which === 39 && snake.dx === 0) {
    snake.dx = grid;
    snake.dy = 0;
  }
  // Стрелка вниз
  else if (e.which === 40 && snake.dy === 0) {
    snake.dy = grid;
    snake.dx = 0;
  }
});
```

5. Запускаем игру. Для этого достаточно запустить предыдущий бесконечный цикл, поэтому пишем:

```
requestAnimationFrame(loop);
```

Полный код можно посмотреть здесь
https://thecode.media/snake-js/