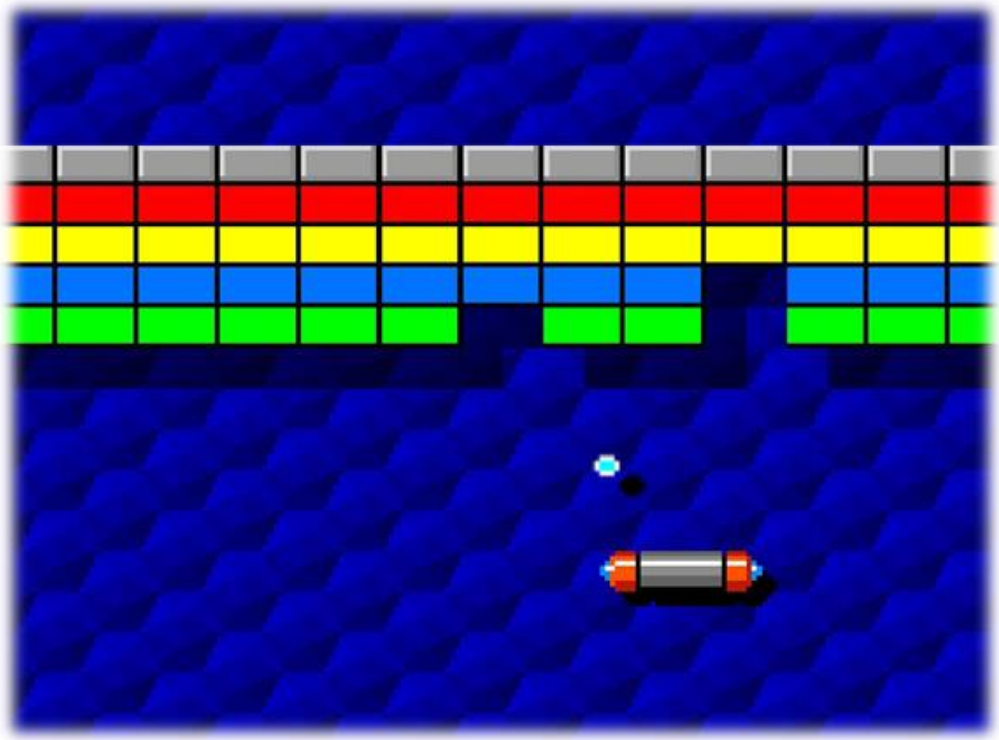

Аркада
на JavaScript

Всё как на старых приставках.

Арканоид — классическая компьютерная игра. Сегодня мы её воссоздадим.

Суть игры такая:

сверху игрового поля стоят несколько рядов кирпичей;

по полю движется шарик, который при касании убирает кирпич и отскакивает в противоположную сторону;

от стен и верха шарик тоже отскакивает;

внизу есть подвижная платформа, как ракетка;

чтобы шарик не упал вниз, игрок двигает платформу влево или вправо, подставляя её под шарик;

если шарик падает мимо платформы — игра останавливается или заканчивается совсем, смотря как запрограммировать;

цель игры — сбить все кирпичи и не дать шарику упасть.

Логика игры

Чтобы было понятно, в какой последовательности мы пишем код, вот общая структура:

Готовим страницу и рисуем на ней игровое поле. Всё остальное будет происходить в скрипте.

Заводим переменные, которые будут отвечать за наполнение уровня, цвет и размер всех игровых элементов.

Сразу помещаем всё, что относится к кирпичам, в одну большую переменную, с которой будем работать всю игру.

Добавляем функцию проверки на касание объектов — вся игра будет строиться на ней.

Делаем главный цикл игры, в котором по очереди сдвигаем все объекты, которые движутся, и убираем те кирпичи, которые нужно убрать. Каждый кадр всё будет отрисовываться заново, чтобы был эффект непрерывного движения.

Готовим страницу

Единственное, что нам здесь понадобится, — сделать стилями чёрный фон, чтобы цветные кирпичи смотрелись на нём эффектнее. Всё остальное берём из шаблона пустой страницы. Пока что на экране будет чернота, рисовать будем дальше.

```
<!DOCTYPE html>
<html>

<head>
  <title>Арканоид</title>
  <style>
    /* стили для всей страницы */
    html, body {
      height: 100%;
      margin: 0;
    }
```

```
 /* отдельные параметры для фона */
    body {
      background: black;
      display: flex;
      align-items: center;
      justify-content: center;
    }
  </style>
</head>

<body>
  <!-- рисуем игровое поле -->
  <canvas width="400" height="500" id="game"></canvas>
  <!-- сам скрипт с игрой -->
  <script>
  </script>
</body>
</html>
```

Переменные

Для игры нам понадобятся две основные переменные (помимо всяких вспомогательных).

Первая переменная отвечает за дизайн уровня — это двумерный массив, где каждая строка отвечает за свой ряд кирпичей. Сами кирпичи сразу обозначаются буквами, которые отвечают за их цвет. Если кирпичей нет, то ряд пустой и там ничего рисоваться не будет. Меняем эту переменную — меняется и внешний вид уровня. В будущем этой переменной может быть много дизайнов уровней.

Вторая переменная — одномерный массив, в котором хранится вся информация обо всех кирпичах: координаты, цвет и размеры каждого кирпича. Именно с этой переменной мы и будем работать всю игру — в самом начале поместим туда все кирпичи из переменной с дизайном, а потом будем убирать их оттуда при касании шарика.

```
// переменная для работы с холстом, на котором будет нарисована игра
const canvas = document.getElementById('game');
const context = canvas.getContext('2d');


// каждый ряд состоит из 14 кирпичей. На уровне будут 6 пустых рядов, а затем 8 рядов с кирпичами
// цвета кирпичей: красный, оранжевый, зелёный и жёлтый
// буква в массиве означает цвет кирпича
const level1 = [
```

```
 [],
  [],
  [],
  [],
  [],
  [],
  ['R','R','R','R','R','R','R','R','R','R','R','R','R','R'],
  ['R','R','R','R','R','R','R','R','R','R','R','R','R','R'],
  ['O','O','O','O','O','O','O','O','O','O','O','O','O','O'],
  ['O','O','O','O','O','O','O','O','O','O','O','O','O','O'],
  ['G','G','G','G','G','G','G','G','G','G','G','G','G','G'],
  ['G','G','G','G','G','G','G','G','G','G','G','G','G','G'],
  ['Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y'],
  ['Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y']
];
```

```
// сопоставляем буквы (R, O, G, Y) с цветами
const colorMap = {
  'R': 'red',
  'O': 'orange',
  'G': 'green',
  'Y': 'yellow'
};

// делаем зазор в 2 пикселя между кирпичами, чтобы отделить их друг от друга
const brickGap = 2;
// размеры каждого кирпича
const brickWidth = 25;
const brickHeight = 12;
```

```
// ширина стены должна занимать оставшееся место на холсте с каждой стороны
// у нас 14 кирпичей по 25 пикселей и 13 промежутков по 2 пикселя, а общая ширина холста — 400 пикселей
// получаем общую ширину стен: 400 − (14 × 25 + 2 × 13) = 24px.
// разделим пополам, чтобы получить ширину каждой стены, и получим 12px
const wallSize = 12;

// платформа, которой управляет игрок
const paddle = {
  // ставим её внизу по центру поля
  x: canvas.width / 2 - brickWidth / 2,
  y: 440,
  // делаем её размером с кирпич
  width: brickWidth,
  height: brickHeight,

  // пока платформа никуда не движется, поэтому направление движения равно нулю
  dx: 0
};
```

```
// шарик, который отскакивает от платформы и уничтожает кирпичи
const ball = {
  // стартовые координаты
  x: 130,
  y: 260,
  // высота и ширина (для простоты это будет квадратный шарик)
  width: 5,
  height: 5,

  // скорость шарика по обеим координатам
  speed: 2,

  // на старте шарик пока никуда не смещается
  dx: 0,
  dy: 0
};
```

```
// основной массив для игры
const bricks = [];

// создадим уровень так: обработаем весь массив level1
// и те места, которые обозначены каким-либо цветом, поместим в игровой массив.
// там же будем хранить координаты начала каждого кирпича и его цвет

// пока у нас есть необработанные элементы в массиве с уровнем — обрабатываем их
for (let row = 0; row < level1.length; row++) {
  for (let col = 0; col < level1[row].length; col++) {

    // находим цвет кирпича
    const colorCode = level1[row][col];
```

```
 // создаём новый элемент игрового массива — с координатами кирпича, цветом, шириной и высотой кирпича
    bricks.push({
      x: wallSize + (brickWidth + brickGap) * col,
      y: wallSize + (brickHeight + brickGap) * row,
      color: colorMap[colorCode],
      width: brickWidth,
      height: brickHeight
    });
  }
}
```

Проверка на касание объектов

Это настолько востребованная функция у разработчиков игр, что проще взять уже готовую, чем писать самому с нуля.

```
// проверка на пересечение объектов
// взяли отсюда: https://developer.mozilla.org/en-US/docs/Games/Techniques/2D_collision_detection
function collides(obj1, obj2) {
  return obj1.x < obj2.x + obj2.width &&
         obj1.x + obj1.width > obj2.x &&
         obj1.y < obj2.y + obj2.height &&
         obj1.y + obj1.height > obj2.y;
}
```

Отслеживаем нажатия на клавиши

Чтобы игрок мог управлять движением платформы, добавим обработчик нажатий на клавиши.

Стрелки влево и вправо двигают платформу в стороны, а пробел запускает шарик, если он не движется. Как только игрок отпускает стрелки, платформа перестаёт двигаться.

```
// отслеживаем нажатия игрока на клавиши
document.addEventListener('keydown', function(e) {
  // стрелка влево
  if (e.which === 37) {
    paddle.dx = -3;
  }
  // стрелка вправо
  else if (e.which === 39) {
    paddle.dx = 3;
  }
```

```
 // обрабатываем нажатие на пробел
  // если шарик не запущен — запускаем его из начальной точки, сверху вниз
  if (ball.dx === 0 && ball.dy === 0 && e.which === 32) {
    ball.dx = ball.speed;
    ball.dy = ball.speed;
  }
});

// как только игрок перестал нажимать клавиши со стрелками — останавливаем платформу
document.addEventListener('keyup', function(e) {
  if (e.which === 37 || e.which === 39) {
    paddle.dx = 0;
  }
});
```

Главный цикл игры

Вся игра — это один большой цикл, где каждое новое положение элементов отрисовывается в новом кадре. Последовательность действий будет такая:

Очищаем игровое поле

Сдвигаем платформу, если было нажатие на стрелки, и следим, чтобы она не улетела за границы стен.

Двигаем шарик со своей скоростью и направлением и смотрим, чтобы он отскакивал от стен и верхней границы.

Если шарик упадёт вниз мимо платформы-ракетки — помещаем его на стартовую позицию и останавливаем, пока не игрок не нажмёт пробел.

Если шарик коснулся платформы — делаем отскок в противоположную сторону.

Когда шарик коснётся кирпича — тоже делаем отскок и сразу убираем этот кирпич.

И только в самом конце мы отрисовываем все игровые элементы на своих новых позициях.

```
// главный цикл игры
function loop() {
  // на каждом кадре — очищаем поле и рисуем всё заново
  requestAnimationFrame(loop);
  context.clearRect(0,0,canvas.width,canvas.height);

  // двигаем платформу с нужной скоростью
  paddle.x += paddle.dx;

  // при этом смотрим, чтобы она не уехала за стены
  if (paddle.x < wallSize) {
    paddle.x = wallSize
  }
  else if (paddle.x + brickWidth > canvas.width - wallSize) {
    paddle.x = canvas.width - wallSize - brickWidth;
  }
```

```
// шарик тоже двигается со своей скоростью
 ball.x += ball.dx;
 ball.y += ball.dy;

 // и его тоже нужно постоянно проверять, чтобы он не улетел за границы стен
 // смотрим левую и правую стенки
 if (ball.x < wallSize) {
   ball.x = wallSize;
   ball.dx *= -1;
 }
 else if (ball.x + ball.width > canvas.width - wallSize) {
   ball.x = canvas.width - wallSize - ball.width;
   ball.dx *= -1;
 }
 // проверяем верхнюю границу
 if (ball.y < wallSize) {
   ball.y = wallSize;
   ball.dy *= -1;
 }
```

```
// перезагружаем шарик, если он улетел вниз, за край игрового поля
  if (ball.y > canvas.height) {
    ball.x = 130;
    ball.y = 260;
    ball.dx = 0;
    ball.dy = 0;
  }


  // проверяем, коснулся ли шарик платформы, которой управляет игрок. Если коснулся — меняем направление движения шарика по оси Y на противоположное
  if (collides(ball, paddle)) {
    ball.dy *= -1;


    // сдвигаем шарик выше платформы, чтобы на следующем кадре это снова не засчиталось за столкновение
    ball.y = paddle.y - ball.height;
  }
```

```
// проверяем, коснулся ли шарик цветного кирпича
// если коснулся — меняем направление движения шарика в зависимости от стенки касания
// для этого в цикле проверяем каждый кирпич на касание
for (let i = 0; i < bricks.length; i++) {
  // берём очередной кирпич
  const brick = bricks[i];

  // если было касание
  if (collides(ball, brick)) {
    // убираем кирпич из массива
    bricks.splice(i, 1);
```

```
 // если шарик коснулся кирпича сверху или снизу — меняем направление движения шарика по оси Y
      if (ball.y + ball.height - ball.speed <= brick.y ||
          ball.y >= brick.y + brick.height - ball.speed) {
        ball.dy *= -1;
      }
      // в противном случае меняем направление движения шарика по оси X
      else {
        ball.dx *= -1;
      }
      // как нашли касание — сразу выходим из цикла проверки
      break;
    }
  }
```

```
// рисуем стены
context.fillStyle = 'lightgrey';
context.fillRect(0, 0, canvas.width, wallSize);
context.fillRect(0, 0, wallSize, canvas.height);
context.fillRect(canvas.width - wallSize, 0, wallSize, canvas.height);

// если шарик в движении — рисуем его
if (ball.dx || ball.dy) {
  context.fillRect(ball.x, ball.y, ball.width, ball.height);
}
```

```
 // рисуем кирпичи
  bricks.forEach(function(brick) {
    context.fillStyle = brick.color;
    context.fillRect(brick.x, brick.y, brick.width, brick.height);
  });

  // рисуем платформу
  context.fillStyle = 'cyan';
  context.fillRect(paddle.x, paddle.y, paddle.width, paddle.height);
}
```

Запускаем игру

Для запуска достаточно одной команды в самом конце скрипта:

requestAnimationFrame(loop);