

16. 캔버스에서 애니메이션 실행하기

객체를 사용해 도형 그리기

① 생성자 함수를 사용해 Circle 객체를 만든다.

```
const canvas = document.querySelector("canvas");
const ctx = canvas.getContext("2d");

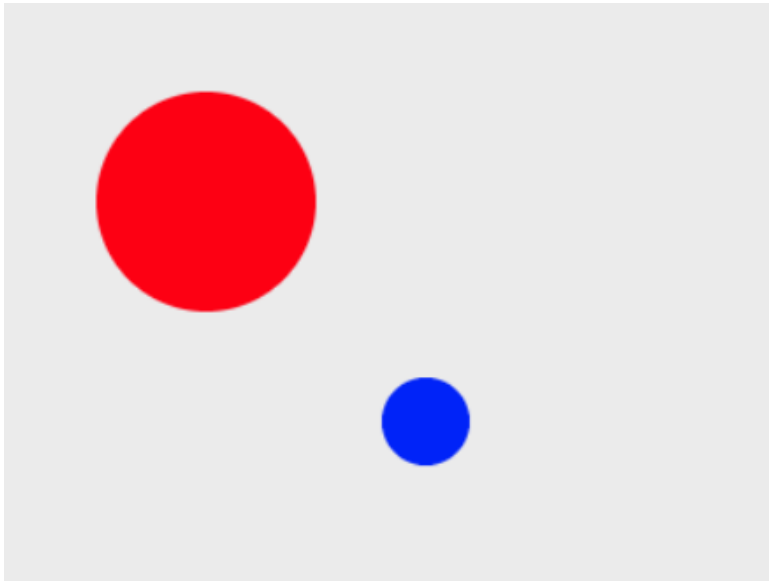
canvas.width = window.innerWidth;    // 캔버스 너비
canvas.height = window.innerHeight;  // 캔버스 높이

function Circle(x, y, radius, color) {
  this.x = x;    // 중점 좌표 x
  this.y = y;    // 중점 좌표 y
  this.radius = radius;    // 반지름
  this.color = color;      // 채우기 색

  // 원을 그리는 draw 메서드
  this.draw = function() {
    ctx.beginPath();
    ctx.fillStyle = this.color;
    ctx.arc(this.x, this.y, this.radius, 0, Math.PI * 2, false);
    ctx.fill();
  }
}
```

② 원의 중심점과 반지름, 색상의 값을 다르게 해서 여러 개의 인스턴스를 만든다.

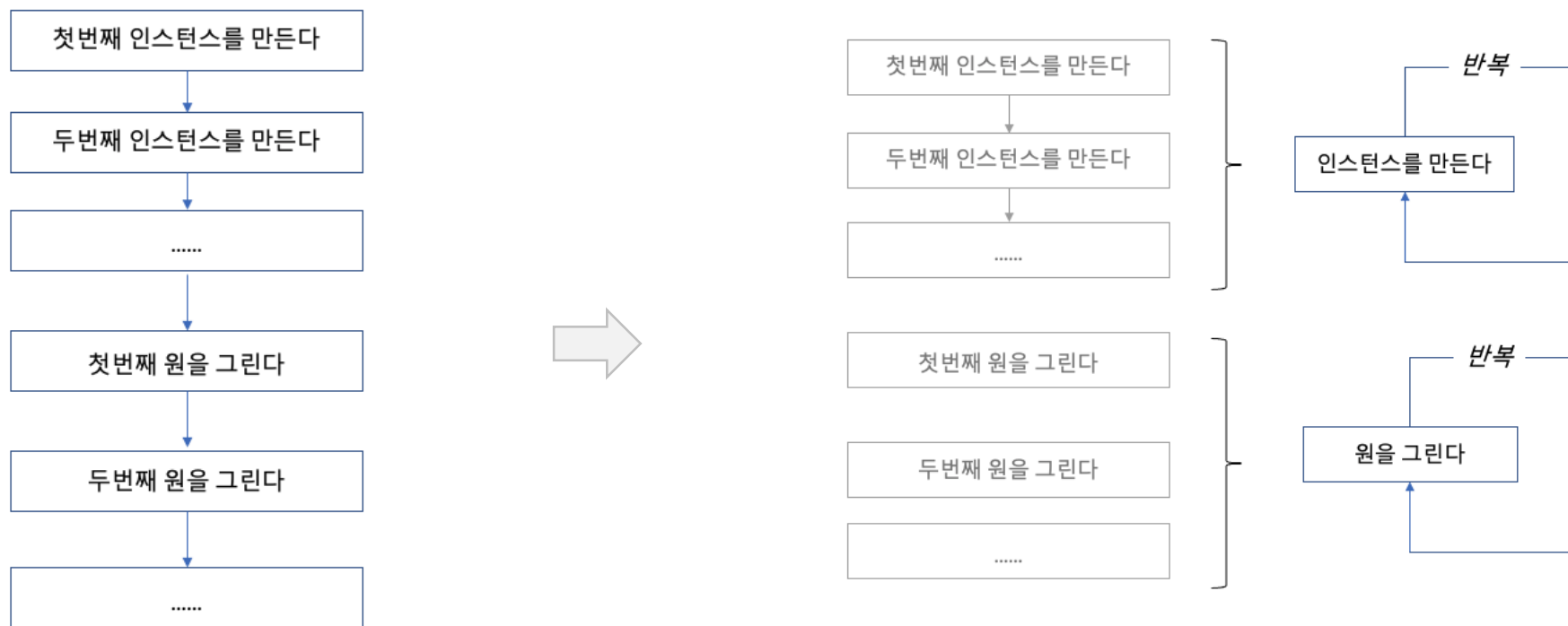
```
const circleOne = new circle(100, 100, 50, "red");  
const circleTwo = new circle(200, 200, 20, "blue");  
circleOne.draw();  
circleTwo.draw();
```



[실습] 화면에 무작위로 여러 개 원 그리기

원의 중심과 크기, 채우기, 색상을 모두 무작위로 선택하도록 선택해서 여러 개의 원을 그려보자.

열 개의 인스턴스를 만들고, 열 개의 원 그리기 명령을 사용 → 좀더 효율적으로 만들려면?



```
const canvas = document.querySelector("canvas");
const ctx = canvas.getContext("2d");

canvas.width = window.innerWidth;    // 캔버스 너비
canvas.height = window.innerHeight;  // 캔버스 높이

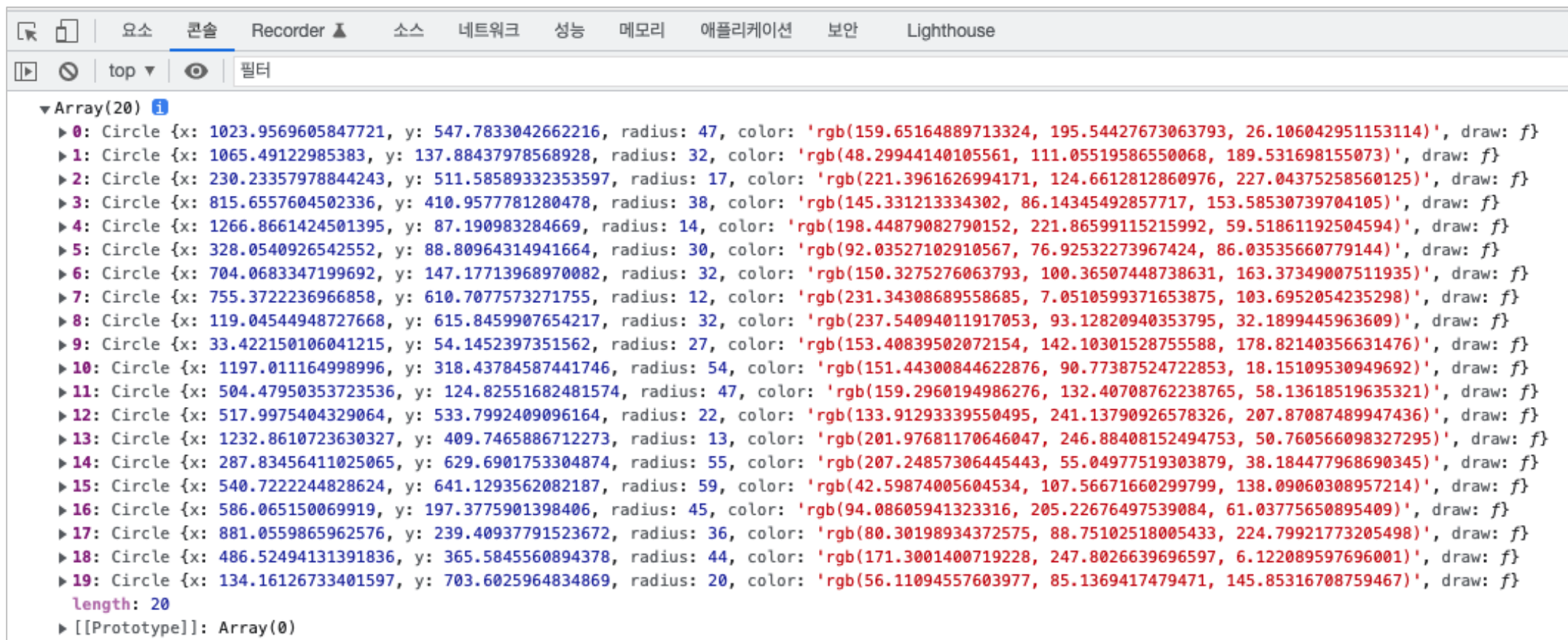
function Circle(x, y, radius, color) {
  this.x = x;    // 중점 좌표 x
  this.y = y;    // 중점 좌표 y
  this.radius = radius;    // 반지름
  this.color = color;    // 채우기 색

  // 원을 그리는 draw 메서드
  this.draw = function() {
    ctx.beginPath();
    ctx.fillStyle = this.color;
    ctx.arc(this.x, this.y, this.radius, 0, Math.PI * 2, false);
    ctx.fill();
  }
}
```

반복문을 사용해 무작위로 20개 원 만들기

```
const objs = [];          // 인스턴스를 저장할 변수
for (let i = 0; i < 20; i++) {
    const radius = Math.floor((Math.random() * 50)) + 10;    // 반지름
    const x = Math.random() * (canvas.width - radius * 2) + radius;    // 원점 x 좌표
    const y = Math.random() * (canvas.height - radius * 2) + radius;    // 원점 y 좌표
    const color = `rgb(${Math.random() * 255}, ${Math.random() * 255}, ${Math.random() * 255})`; // 색상
    objs.push(new Circle(x, y, radius, color));    // objs에 인스턴스를 추가합니다.
}
console.log(objs);
```

콘솔 창에서 20개의 인스턴스 객체가 만들어졌는지 확인한다.



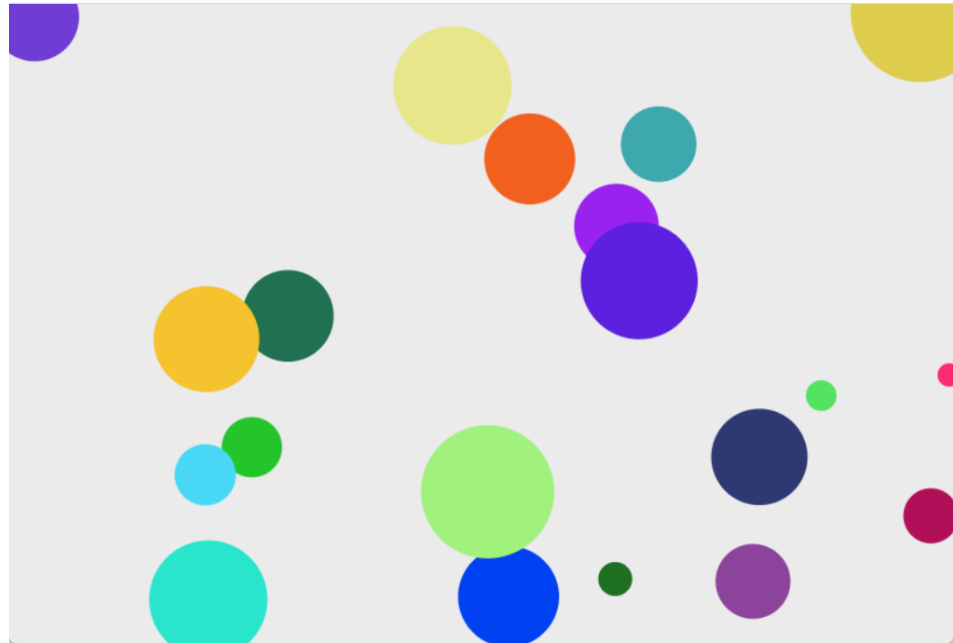
The screenshot shows a web browser's developer console with the 'Console' tab selected. The console displays an array of 20 objects, each representing a 'Circle'. Each object has properties for 'x', 'y', 'radius', 'color', and 'draw'. The 'x' and 'y' coordinates are long decimal values. The 'radius' values are integers ranging from 12 to 59. The 'color' values are strings in 'rgb(r, g, b)' format with decimal values. The 'draw' property is a function 'f'. The array is labeled 'Array(20)' and has a 'length' of 20. Below the array, the prototype chain is shown as '[[Prototype]]: Array(0)'.

```
▼ Array(20) ⓘ
  ► 0: Circle {x: 1023.9569605847721, y: 547.7833042662216, radius: 47, color: 'rgb(159.65164889713324, 195.54427673063793, 26.106042951153114)', draw: f}
  ► 1: Circle {x: 1065.49122985383, y: 137.88437978568928, radius: 32, color: 'rgb(48.29944140105561, 111.05519586550068, 189.531698155073)', draw: f}
  ► 2: Circle {x: 230.23357978844243, y: 511.58589332353597, radius: 17, color: 'rgb(221.3961626994171, 124.6612812860976, 227.04375258560125)', draw: f}
  ► 3: Circle {x: 815.6557604502336, y: 410.9577781280478, radius: 38, color: 'rgb(145.331213334302, 86.14345492857717, 153.58530739704105)', draw: f}
  ► 4: Circle {x: 1266.8661424501395, y: 87.190983284669, radius: 14, color: 'rgb(198.44879082790152, 221.86599115215992, 59.51861192504594)', draw: f}
  ► 5: Circle {x: 328.0540926542552, y: 88.80964314941664, radius: 30, color: 'rgb(92.03527102910567, 76.92532273967424, 86.03535660779144)', draw: f}
  ► 6: Circle {x: 704.0683347199692, y: 147.17713968970082, radius: 32, color: 'rgb(150.3275276063793, 100.36507448738631, 163.37349007511935)', draw: f}
  ► 7: Circle {x: 755.3722236966858, y: 610.7077573271755, radius: 12, color: 'rgb(231.34308689558685, 7.0510599371653875, 103.6952054235298)', draw: f}
  ► 8: Circle {x: 119.04544948727668, y: 615.8459907654217, radius: 32, color: 'rgb(237.54094011917053, 93.12820940353795, 32.1899445963609)', draw: f}
  ► 9: Circle {x: 33.422150106041215, y: 54.1452397351562, radius: 27, color: 'rgb(153.40839502072154, 142.10301528755588, 178.82140356631476)', draw: f}
  ► 10: Circle {x: 1197.011164998996, y: 318.43784587441746, radius: 54, color: 'rgb(151.44300844622876, 90.77387524722853, 18.15109530949692)', draw: f}
  ► 11: Circle {x: 504.47950353723536, y: 124.82551682481574, radius: 47, color: 'rgb(159.2960194986276, 132.40708762238765, 58.13618519635321)', draw: f}
  ► 12: Circle {x: 517.9975404329064, y: 533.7992409096164, radius: 22, color: 'rgb(133.91293339550495, 241.13790926578326, 207.87087489947436)', draw: f}
  ► 13: Circle {x: 1232.8610723630327, y: 409.7465886712273, radius: 13, color: 'rgb(201.97681170646047, 246.88408152494753, 50.760566098327295)', draw: f}
  ► 14: Circle {x: 287.83456411025065, y: 629.6901753304874, radius: 55, color: 'rgb(207.24857306445443, 55.04977519303879, 38.184477968690345)', draw: f}
  ► 15: Circle {x: 540.722244828624, y: 641.1293562082187, radius: 59, color: 'rgb(42.59874005604534, 107.56671660299799, 138.09060308957214)', draw: f}
  ► 16: Circle {x: 586.065150069919, y: 197.3775901398406, radius: 45, color: 'rgb(94.08605941323316, 205.22676497539084, 61.03775650895409)', draw: f}
  ► 17: Circle {x: 881.0559865962576, y: 239.40937791523672, radius: 36, color: 'rgb(80.30198934372575, 88.75102518005433, 224.79921773205498)', draw: f}
  ► 18: Circle {x: 486.52494131391836, y: 365.5845560894378, radius: 44, color: 'rgb(171.3001400719228, 247.8026639696597, 6.122089597696001)', draw: f}
  ► 19: Circle {x: 134.16126733401597, y: 703.6025964834869, radius: 20, color: 'rgb(56.11094557603977, 85.1369417479471, 145.85316708759467)', draw: f}
  length: 20
  [[Prototype]]: Array(0)
```


console.log() 문을 주석처리하거나 삭제한 후 인스턴스 객체를 화면에 표시하는 소스 추가

```
// console.log(objs);  
  
for (let i = 0; i < objs.length; i++) {  
  objs[i].draw();  
}
```

새로고침할 때마다 무작위로 원이 그려진다.



애니메이션 만들기

그래픽 요소의 좌표 옮기기

가장 간단한 애니메이션은 요소의 좌표를 옮겨 요소를 움직이는 것.

그래픽 객체를 만든 후 가로로 일정한 크기만큼 움직이는 소스를 작성해 보자

1) 원을 그리기 위해 circle 객체 정의

```
const canvas =  
document.querySelector("canvas");  
const ctx = canvas.getContext("2d");  
  
canvas.width = window.innerWidth;  
canvas.height = window.innerHeight;  
  
const circle = {  
  x: 100,  
  y: 100,  
  radius: 30,  
  dx: 4,    // 가로로 움직일 크기  
  dy: 4,    // 세로로 움직일 크기  
  color: "#222"  
}
```

2) 화면에 원 그리는 drawCircle 함수와 가로로 dx만큼 움직이는 move 함수 정의

```
function drawCircle() {  
    ctx.beginPath();  
    ctx.arc(circle.x, circle.y, circle.radius, 0, Math.PI * 2, false);  
    ctx.fillStyle = circle.color;  
    ctx.fill();  
}
```

```
function move() {  
    drawCircle();  
    circle.x += circle.dx;  
}
```

dx만큼 이동한 후 다시 화면에 그려야 한다.

→ move() 함수 반복

```
move();
```

window.requestAnimationFrame()

애니메이션은 한 위치에서 다른 위치로 옮겨가면서 계속 그래픽 요소를 화면에 그려야 한다.
즉, 좌표를 옮기고 그래픽 요소를 그리는 함수를 반복.

반복 애니메이션을 위해 requestAnimationFrame() 메서드 사용.

```
requestAnimationFrame(콜백)
```

requestAnimationFrame() 메서드에 반복할 함수를 지정하면 계속 반복한다.

기존의 move() 함수 부분을 다음과 같이 지정하자

```
function move() {  
    drawCircle();  
    circle.x += circle.dx;  
    requestAnimationFrame(move);    // move 함수 반복.  
}
```

3) 웹 브라우저에서 확인하기

(50, 50) 위치에 있던 원이 (54, 50), (58, 50),과 같은 방식으로 원점 좌표가 바뀌면서 움직인다.



이동 과정이 한꺼번에 그려지기 때문에 마치 직선처럼 보인다.

그리고 끝나는 위치가 정해져 있지 않아서 끝없이 가로로 그려진다.

[실습] 왔다갔다 움직이는 원 만들기

앞에서 좌표만 움직였던 애니메이션은 마치 직선처럼 표시된다.

원 형태가 유지되면서 원이 움직이는 애니메이션을 만들려면?

→ `translate()` 에서처럼 캔버스의 원점을 옮기고 도형을 그려야 한다.

- ① 시작 위치에서 도형을 그린다.
- ② 캔버스를 지웁니다. 보통 `clearRect()` 메서드를 사용해 캔버스 크기만큼 지운다.
- ③ 저장해야 할 스타일이 있으면 `save()` 메서드를 사용해 캔버스 상태를 저장한다.
- ④ 새로운 위치에 도형을 그린다.
- ⑤ 저장한 캔버스 상태가 있으면 저장한 상태를 복구한다.

앞에서 만들어 놓은 move.js 파일에 이어서 연습해 보자

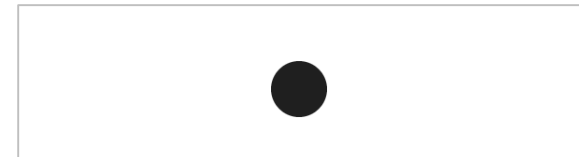
원이 가로로 이동하는 것을 애니메이션으로 만들려면 위치를 옮기고, 캔버스를 지운 후 원을 그려야 한다.

move() 함수를 다음과 같이 수정한다

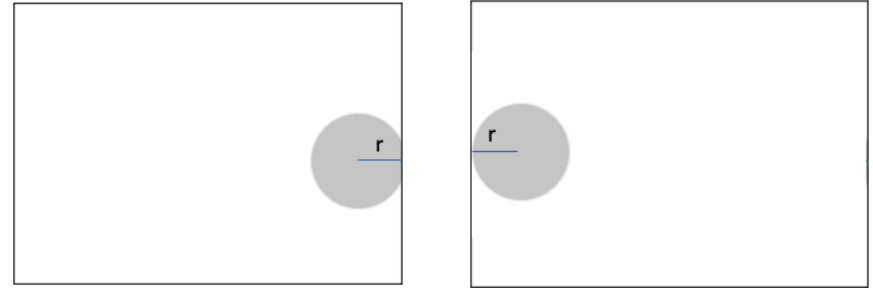
```
function move() {  
  ctx.clearRect(0, 0, canvas.width, canvas.height);  
  drawCircle();  
  
  circle.x += circle.dx;    // circle.x = circle.x + circle.dx  
  
  requestAnimationFrame(move);  
}
```

웹 브라우저에서 확인하기

원이 오른쪽 방향으로 움직이다가 결국에는 캔버스 영역을 벗어남



원이 움직이다가 캔버스 영역의 왼쪽이나 오른쪽 끝에 닿으면 반대 방향으로 움직이게 해보자

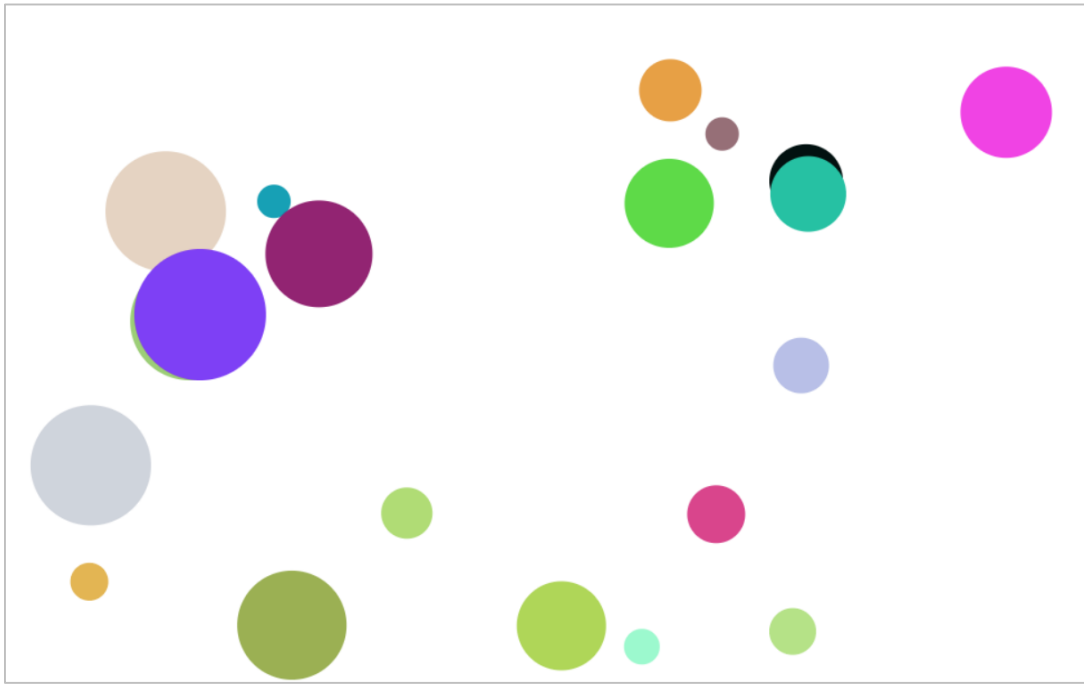


move() 함수를 다음과 같이 수정한다

```
function move() {  
  ctx.clearRect(0, 0, canvas.width, canvas.height);  
  drawCircle();  
  
  circle.x += circle.dx;  
  
  if (circle.x + circle.radius > canvas.width || circle.x - circle.radius < 0)  
  {  
    circle.dx = -circle.dx;    // 방향 바꾸기  
  }  
  
  requestAnimationFrame(move);  
}
```

[실습] 사방으로 통통튀는 애니메이션 만들기

16Wanimation-3.html을 웹 브라우저에서 열면 20개의 원이 그려진다.



원들이 사방으로 움직이는 애니메이션을 만들어 보자

애니메이션을 실행하는 update() 함수 :

캔버스를 지우고 objs 배열에 저장된 20개의 원에 각각 애니메이션을 적용해야 한다.

1) 기존 소스에서 원을 그리는 for문을 삭제하고, update() 함수 추가

```
.....  
for (let i = 0; i < objs.length; i++) {  
  objs[i].draw();  
}  
function update() {  
  ctx.clearRect(0, 0, canvas.width, canvas.height);  
  for (let i = 0; i < objs.length; i++) {  
    let obj = objs[i];  
    obj.animate(); animate() 메서드를 정의해야 한다  
  }  
  requestAnimationFrame(update);  
}
```

update();

2) 무작위로 움직이기 위해 dx, dy를 무작위수로 지정한다.

```
function Circle(x, y, radius, color) {  
    this.x = x;  
    this.y = y;  
    this.radius = radius;  
    this.color = color;  
  
    this.dx = Math.floor(Math.random() * 4) + 1;  
    this.dy = Math.floor(Math.random() * 4) + 1;  
  
    .....  
}
```

3) 원을 움직이는 animate() 메서드를 정의한다.

```
function Circle(x, y, radius, color) {  
    .....  
    this.animate = function() {  
        this.x += this.dx;  
        this.y += this.dy;  
  
        if (this.x + this.radius > canvas.width || this.x - this.radius < 0) {  
            this.dx = -this.dx;  
        }  
        if (this.y + this.radius > canvas.height || this.y - this.radius < 0) {  
            this.dy = -this.dy;  
        }  
        this.draw();  
    }  
}
```

