**동기/비동기 프로그래밍 2**

**async / await 예제 문제 ~ json 데이터 가져와서 화면에 출력해 보기**

[HTML]

<body>

  <h1>회원정보</h1>

  <div id="result"></div>

  <script src="/script1.js"></script>

</body>

[CSS]

\* {

  margin: 0;

  padding: 0;

}

table {

  width: 300px;

  margin: 10px;

  display: inline-block;

  border-collapse: collapse;

}

th {

  width: 80px;

  border: 1px solid #ccc;

}

td {

  border: 1px solid #ccc;

  width: 210px;

  padding: 10px;

}

[JS]

async function init() {

  const response = await fetch("https://jsonplaceholder.typicode.com/users"); //JSON 파일 주소 복사해서 가져오기

  const users =  await response.json();

  display(users) // 함수호출

}

function display(users) {

  const result = document.querySelector("#result");

  let string = "";

  // forEach 각각의 데이터를 넣기위해!

  users.forEach((user) => {

    string += `

    <table>

    <tr><th>이름</th><td>${user.name}</td></tr>

    <tr><th>아이디</th><td>${user.username}</td></tr>

    <tr><th>이메일</th><td>${user.email}</td></tr>

    </table>

    `;

  });

  result.innerHTML = string;

}

init()  // 함수호출

**캔버스**

**소스코드를 활용해서 그림을 그릴 수 있는 기능 부여!**

* 자바스크립트를 활용해서 그림을 그릴 수 있다.
* 에니메이션 제작 및 생성이 가능하다.

1. 자바스크립트에서 캔버스 기능을 사용하려면 HTML **<canvas>태그**가 필요
2. 컨텍스트를 설정해야한다  
   > 메서드 fillStyle, fillRect  
   > 컨텍스트 개념 : 프로그램에 특정 실행문을 호출하고 응답 받을 수 있는 준비를 해놓은 상태를 컨텍스트라고 한다.
3. 컴퓨터 프로그래밍 개발 시, 사용하는 각도 개념 = radian (일상생활에선 degree..  
   > radian = 180 / 파이(\*원주율)  
   > 1도 = 파이 / 180 !!!!!!!!  
   > 자바스크립트에서 파이는 Math.PI  
   > 자바스크립트에서 내가 60도를 움직이고 싶을 땐? (Math.PI / 180) \* 60 하면 됌  
   **90도 = Math.PI \* 0.5**

**180도 = Math.PI**

**270도 = Math.PI \* 1.5**

**360도 = 0도 = Math.PI \* 2**

. const canvas = document.querySelector("canvas");

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

ctx.fillStyle = "rgb(200, 0, 0)";

실제로 들어가는 색상 지정

ctx.fillRect(10, 10, 50, 100);

캔버스 위치 설정하긔 ..

캔버스 사이즈 화면 전체로 설정하기

canvas.width = window.innerWidth;

canvas.height= window.innerHeight;

사각형을 그려보자!

\* 사각형을 그리는 메서드 .........

1. fillRect(x, y, width, height) : 색상을 채우는 사각형을 만들 때

2. strokRect(x, y, width, height) : 테두리만 있는 사각형을 만들 때

3. clearRect(x, y, width, height) : 특정 위치 및 사이즈의 사각영역을 지울 때

==================== 사각형 ====================

const canvas = document.querySelector("canvas") // 기본세팅

const ctx = canvas.getContext("2d") // 필수사항!

ctx.fillStyle = "rgb(200, 0, 0)";

ctx.fillRect(10, 10, 200, 100);

ctx.strokeStyle = "blue"

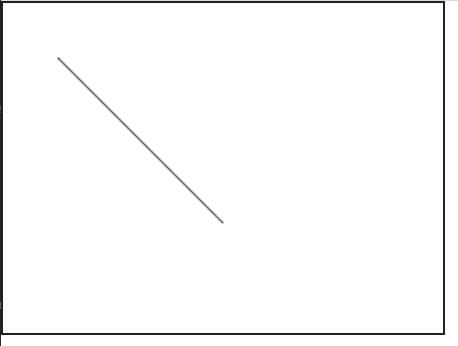
ctx.strokeRect(10, 10, 200, 100);

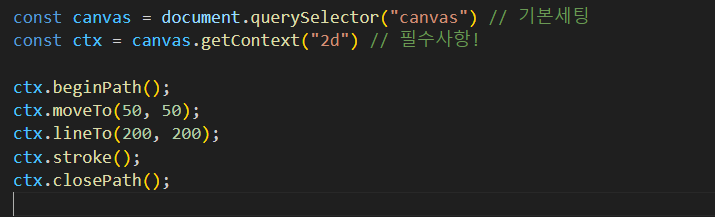
ctx.fillStyle = "rgba(0, 0, 200, 0.5)"; // 반투명!

ctx.fillRect(50, 50, 120, 100);

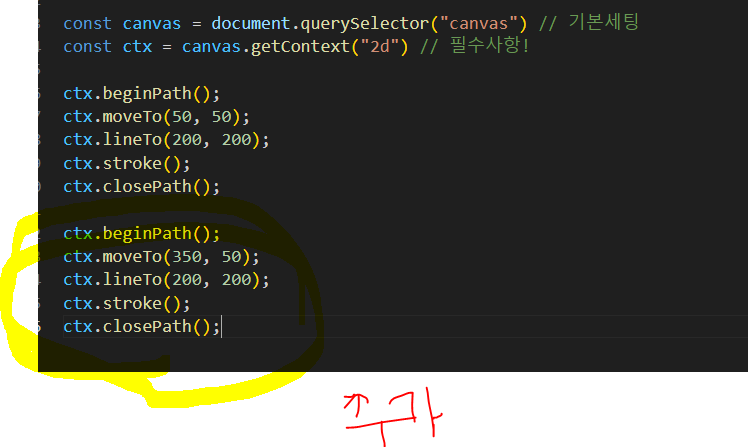
ctx.clearRect(70, 80, 80, 45); // 삭제하긔

삼각형을 그려보자!

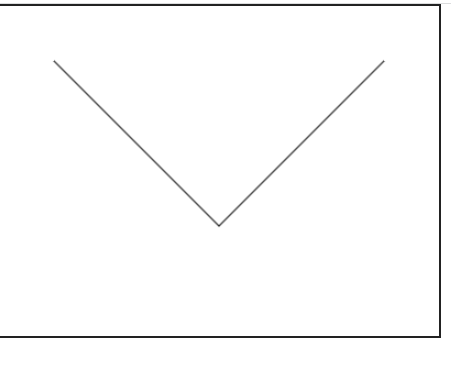


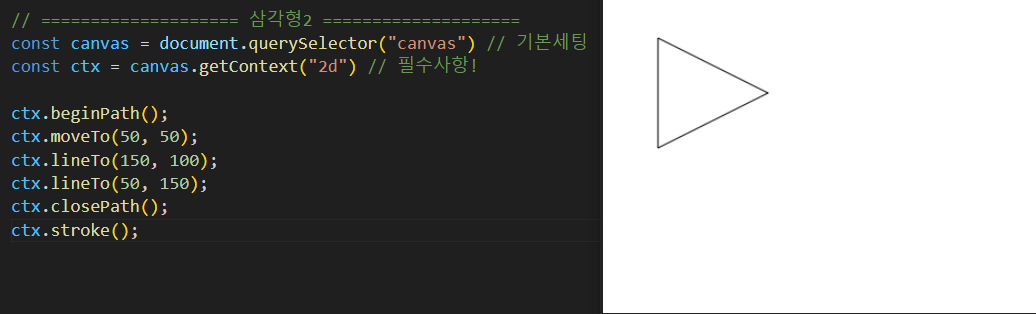


여기서 선을 이으고 싶다면

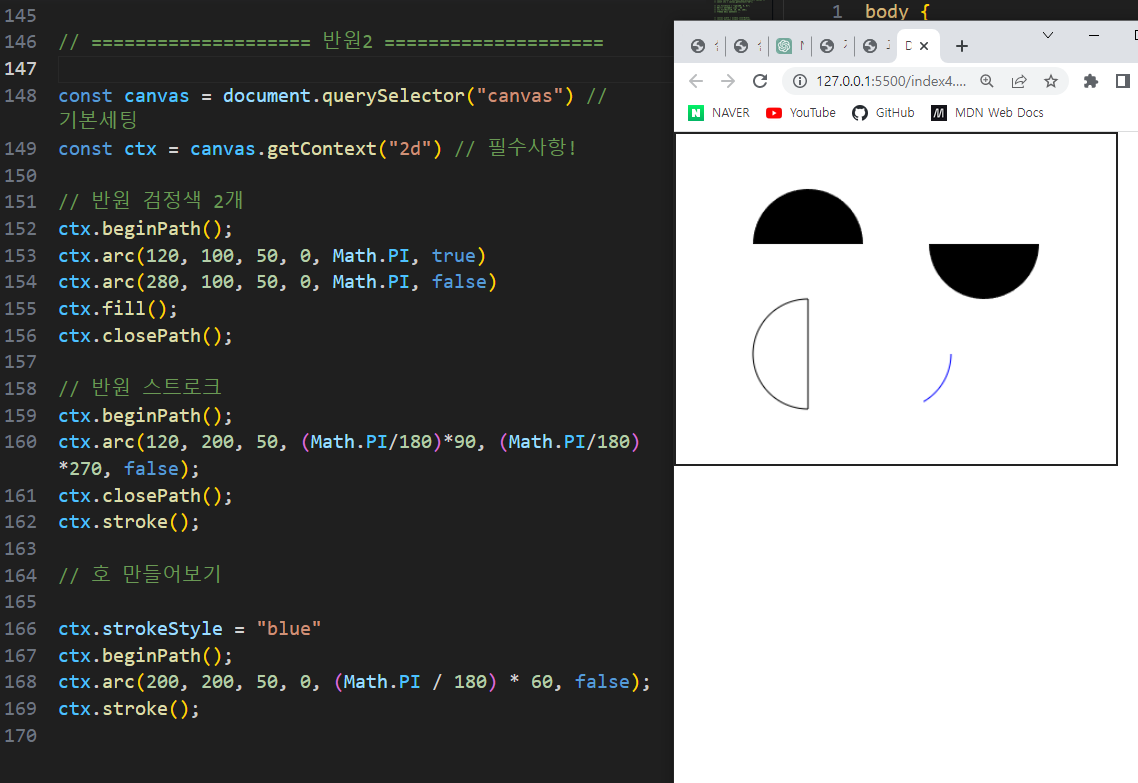


결과

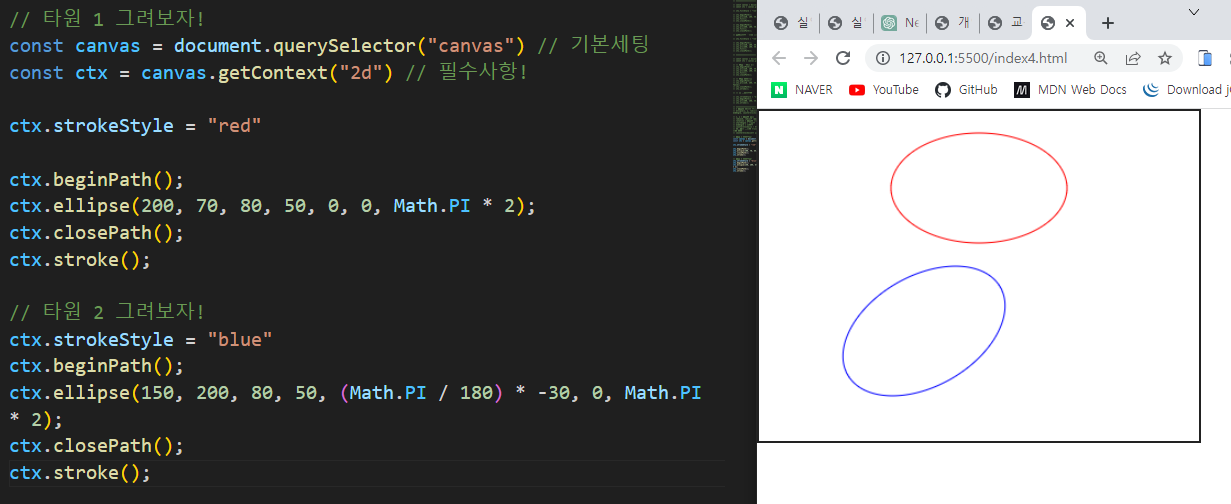




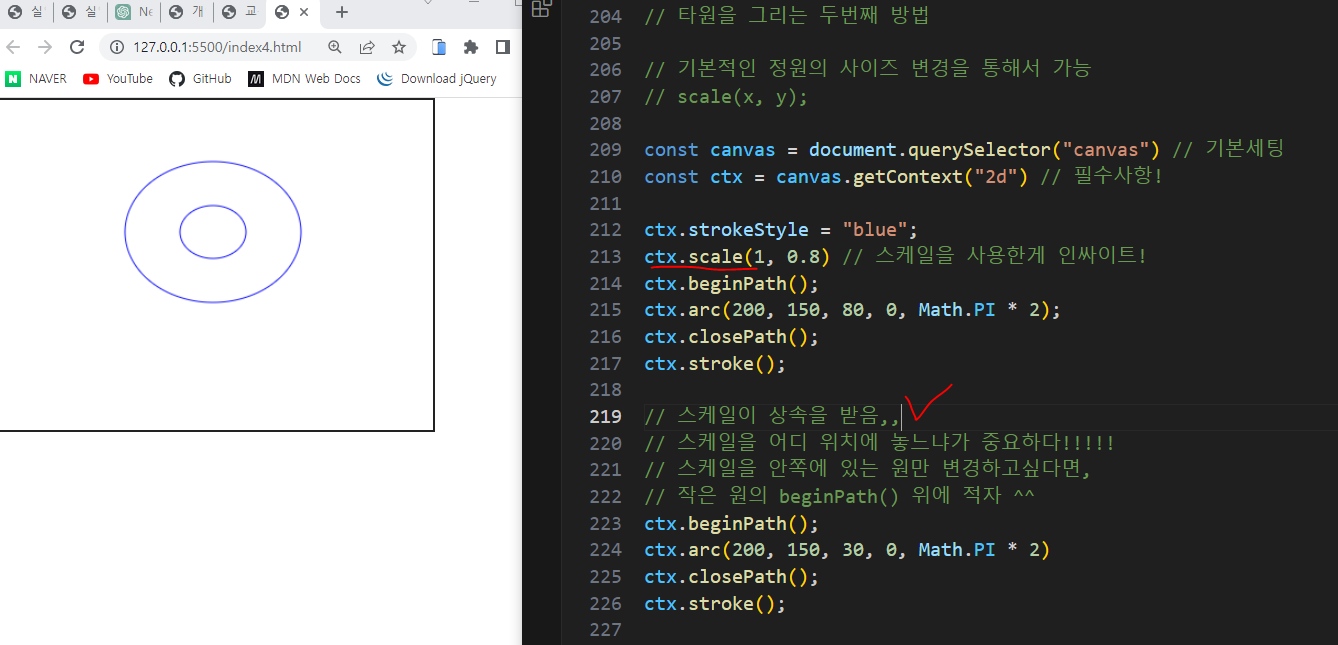
**원 그리기**



**타원 그리기**



**타원을 그리는 두번째 방법**



곡선 그리기

\* 곡선을 그리는 메서드

: 베지에 Bezier

포토샵에서 펜툴 생각하기!! 펜툴 쓸 때 시작점 말고 다른곳 클릭시 2번째 클릭하는 것 부터가 조절점이라고 생각하면 된다.

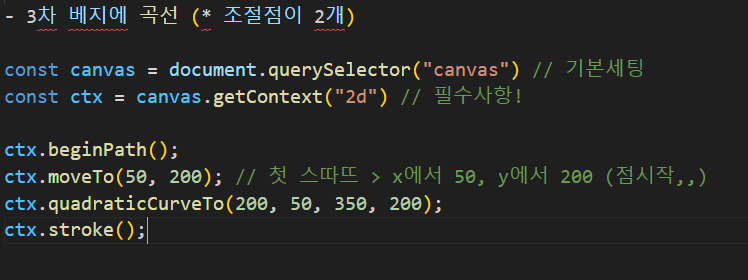
- 2차 베지에 곡선 (\*조절점이 1개)

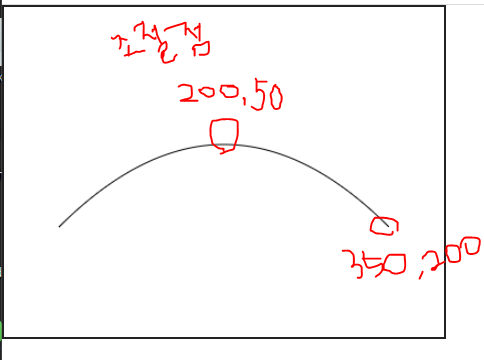
quadraticCurveTo > [ cpx, cpy, x, y ]

cpx, cpy : 조절점 좌표

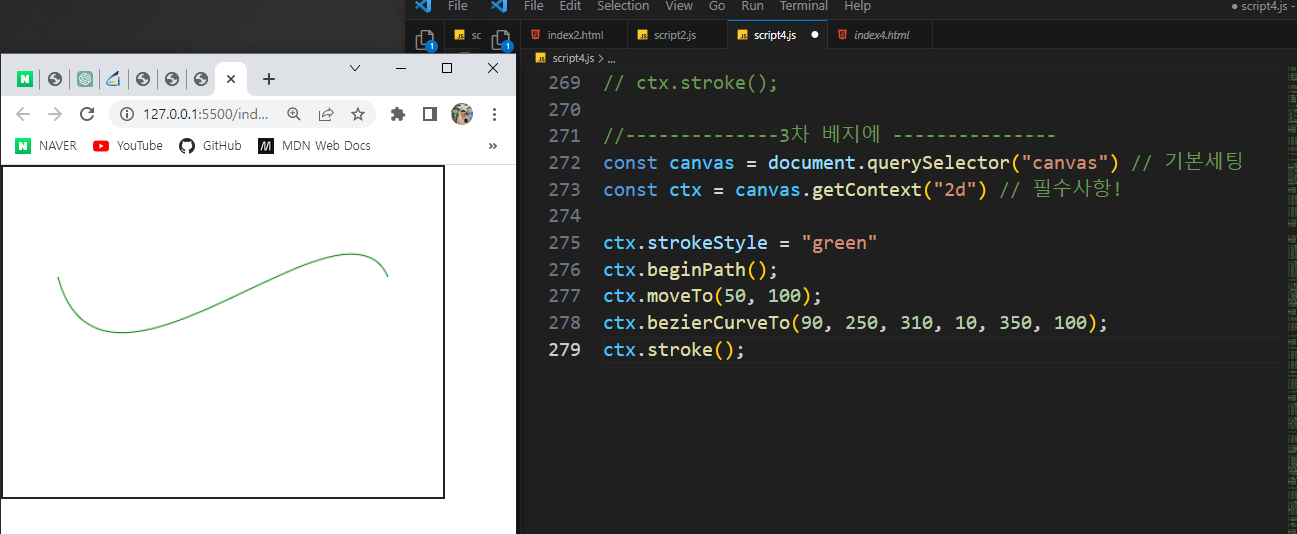
x, y : 곡선이 끝나는 좌표

- 3차 베지에 곡선 (\* 조절점이 2개)

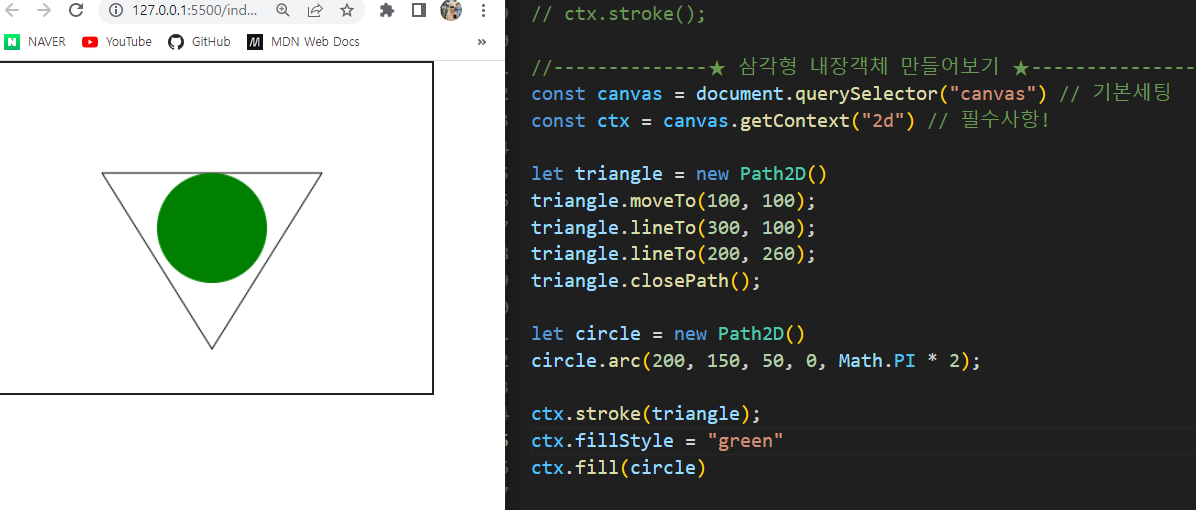




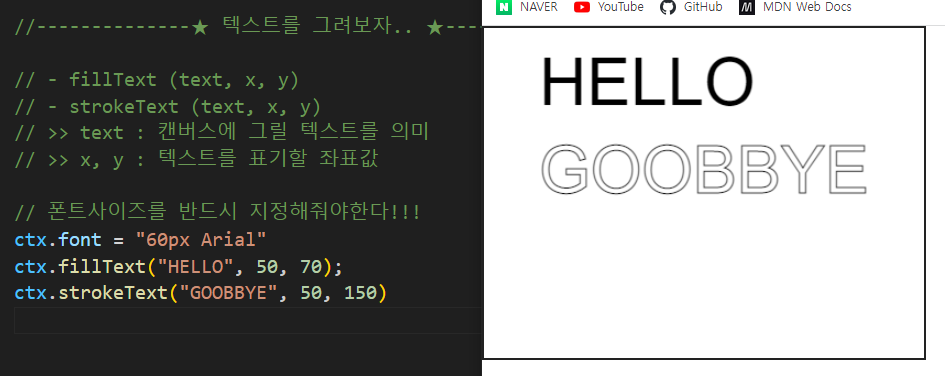
3차 베지에 그려보기



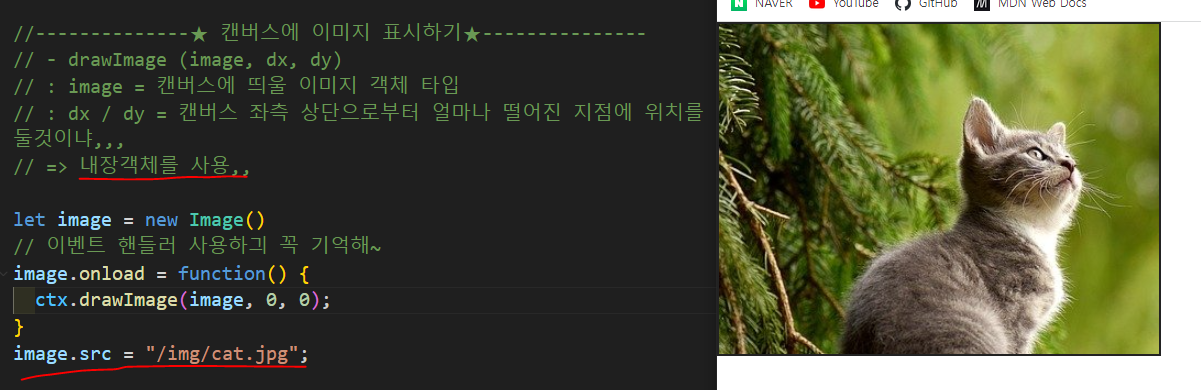
내장객체 활용해서 도형 만들기



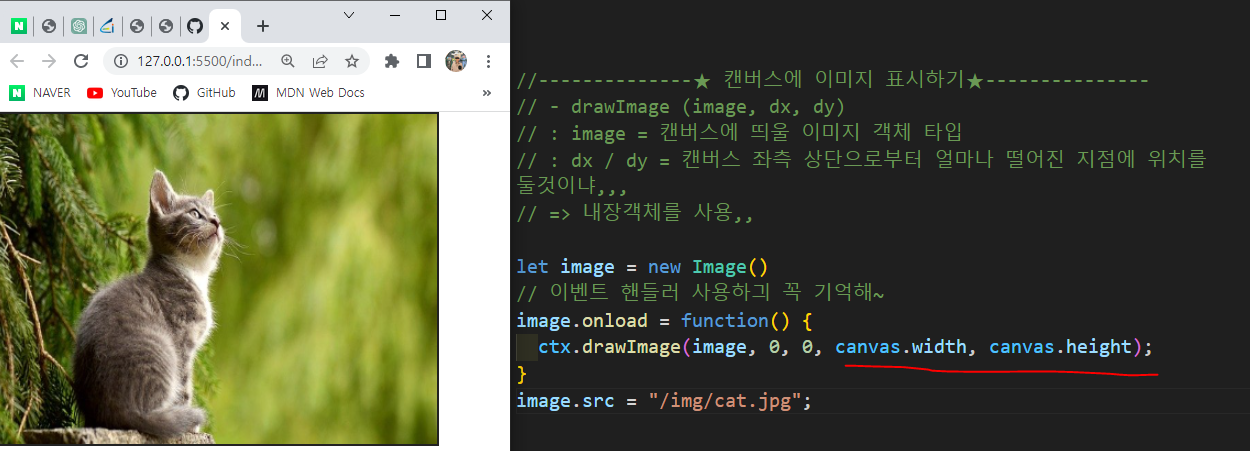
텍스트를 그려보자



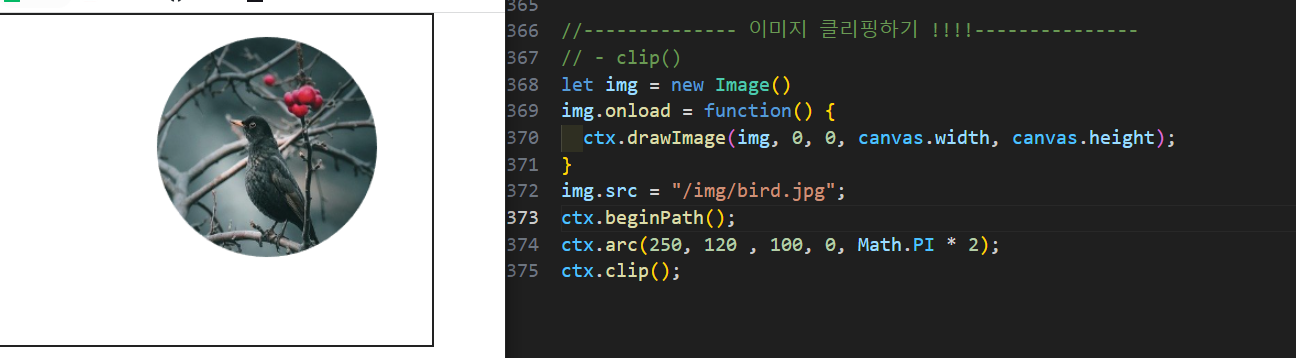
이미지 내장객체를 활용하여 불러오기



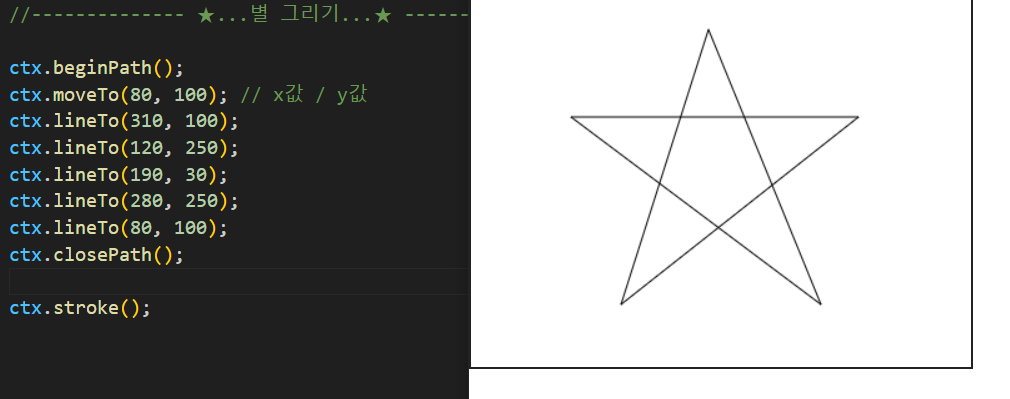
캔버스에 꽉 차게 사이즈 조정하기



클리핑 마스크 만들기



별 그리기



고양이 클리핑 마스크 만들기



23 05 01

캔버스 이어서!

메서드

const canvas = document.querySelector("canvas");

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

canvas.width = window.innerWidth;

canvas.height = window.innerHeight;

// ctx.globalAlpha = 0.3;

// globalAlpha > 0에 가까울수록 투명해짐!

ctx.fillStyle = "rgb(255, 0, 0, 0.2)"

ctx.fillRect(50, 50, 100, 50);

ctx.fillStyle = "rgb(0, 0, 255, 0.4)"

ctx.fillRect(150, 50, 100, 50);

ctx.fillStyle = "rgb(0,255, 0, 0.8)"

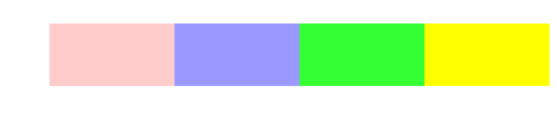
ctx.fillRect(250, 50, 100, 50);

ctx.fillStyle = "rgb(255, 255, 0, 1)"

ctx.fillRect(350, 50, 100, 50);

// fillStyle에서 마지막 숫자는 투명도를 조절할 수 있다!

// 1이 제일 투명도가 없는 것



1. 선형 그라데이션

> createLinearGradient(x1, y1, x2, y2);

- 색 중지점 할당 메서드 > addColorStop(position, color); 정의 필요!

let linGrad = ctx.createLinearGradient(0, 0, 0, 200);

linGrad.addColorStop(0, "#000");

linGrad.addColorStop(0.6, "#fff");

linGrad.addColorStop(1, "#eee");

ctx.fillStyle = linGrad;

ctx.fillRect(0, 0, 100, 200);

2. 원형 그라데이션

> createRadialGradient(x1, y1, r1, x2, y2, r2);

let radGrad = ctx.createRadialGradient(55, 60, 10, 80, 90, 100);

radGrad.addColorStop(0, "white")

radGrad.addColorStop(0.4, "yellow")

radGrad.addColorStop(1, "orange")

ctx.fillStyle = radGrad;

ctx.beginPath();

ctx.arc(100, 100, 80, 0, Math.PI \* 2, false); // 반원이면 true로 설정!> 반시계반향으로

ctx.fill();

// 3. 패턴 채우기

// : createPattern(Image, type);

let img = new Image();

img.onload = function() {

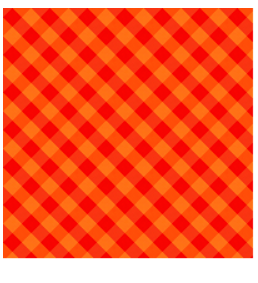
let pattern = ctx.createPattern(img, "repeat");

ctx.fillStyle = pattern;

ctx.fillRect(0, 0, 200, 200);

}

img.src = "/img/pattern.png"



5. 텍스트 선 관련 스타일 속성 :

- 선 굵기 속성 : ctx.lineWidth = 값

- 선의 끝 모양 : ctx.lineCap = butt || round || square (\*선 양쪽 끝에 높이가 선 너비의 1/2인 사각형을 추가!)

- 선 교차 스타일 : lineJoin = bevel || miter || round

const lineCap = ["butt", "round", "square"];

ctx.strokeStyle = "#222"

for문 활용해서 캡 씌우고 3개 연속 만들기

for (let i = 0; i < lineCap.length; i++) {

ctx.lineWidth = 15;

ctx.lineCap = lineCap[i];

ctx.beginPath();

ctx.moveTo(50, 50 + i \* 30);

ctx.lineTo(350, 50 + i \* 30);

ctx.stroke(); // 이게 있어야 선이 그려짐..!

}

// 선 교차 스타일 예습 !!!!!

const lineJoin = ["bevel", "miter", "round"];

for (let i = 0; i < lineJoin.length; i++) {

ctx.lineWidth = 20;

ctx.lineJoin = lineJoin[i];

ctx.beginPath()

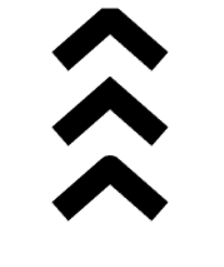
ctx.moveTo(60, 60 \* i + 50);

ctx.lineTo(100, 60 \* i + 15);

ctx.lineTo(140, 60 \* i + 50);

ctx.stroke();

}



// translate 사용해보기! 위치 자체를 옮기기

canvas.width = window.innerWidth;

canvas.height = window.innerHeight;

ctx.fillStyle = "#ccc";

ctx.fillRect(10, 10, 100, 100);

ctx.save();

ctx.translate(150, 150);

ctx.fillStyle = "#222";

ctx.fillRect(10, 10, 100, 100);

// save와 restore은 쌍둥이처럼 움직인다!

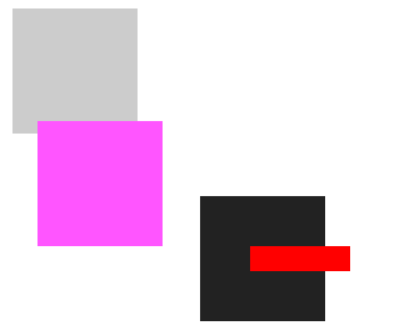
ctx.fillStyle = "red";

ctx.fillRect(50, 50, 80, 20);

ctx.restore(); //복구하라는 뜻

ctx.fillStyle = "#f5f";

ctx.fillRect(30, 100, 100, 100);



23.05.02

==================================

중첩된 (겹쳐진) 도형 중에 깔려있는 걸 destination이라고한다

ctx.fillStyle = "#ccc";

ctx.fillRect(100, 50, 100, 100);

ctx.globalCompositeOperation = "xor"

// 중첩된 (겹쳐진) 도형 중에 위에 있는걸 source라고한다

ctx.fillStyle = "#222"

ctx.beginPath();

ctx.arc(180, 120, 50, 0, Math.PI \* 2, false);

ctx.fill();

.globalCompositeOperation 타입 종류!

★ source ★

1) source-over : source로 시작하는 그래픽 요소를 destination

요소 위쪽에 배치시킴 > 그냥 디폴트값임 어짜피 소스는 맨밑에 있는거니깐

2) source-in : source를 그리면서 destination 요소와

겹쳐지는 부분만 나오게! 그리고 나머지는 투명하게 처리 > 교집합 부분만

3) source-out : source를 그리면서 destination 요소와

겹쳐지지 않는 부분만 나오게합니다 > 소스에서 겹쳐지지 않는 부분만!

4) source-atop : source를 그리면서 destination 요소와

겹쳐지는 부분을 그리고, 그외 destination은 불투명하게 처리

★ destination ★

1) destination-over : destination로 시작하는 그래픽 요소를 source

요소 위쪽에 배치시킴 > 그냥 디폴트값임 어짜피 소스는 맨밑에 있는거니깐

2) destination-in : destination를 그리면서 source 요소와

겹쳐지는 부분만 나오게! 그리고 나머지는 투명하게 처리 > 교집합 부분만

3) destination-out : destination를 그리면서 source 요소와

겹쳐지지 않는 부분만 나오게합니다 > 소스에서 겹쳐지지 않는 부분만!

4) destination-atop : destination를 그리면서 source 요소와

겹쳐지는 부분을 그리고, 그외 source은 불투명하게 처리

==================================

복습 차원으로 실습 해보자 ★

글자에 꽃 이미지 클리핑마스크 넣어보긔

ctx.beginPath();

ctx.font = "bold 200px sans-serif"

ctx.fillText("GOOD", 30, 320);

ctx.globalCompositeOperation = "source-atop"

ctx.beginPath();

let img = new Image();

img.onload = function() {

ctx.drawImage(img, 0, 0, 800, 500);

}

img.src = "/img/flower.jpg"

================== ★ 본격적인 에니메이션 실습 1 ★ ================

function Circle(x, y, radius, color) {

this.x = x;

this.y = y;

this.radous = radius;

this.color = color;

this.draw = function() {

ctx.beginPath();

ctx.fillStyle = this.color

ctx.arc(this.x, this.y, this.radous, 0, Math.PI \* 2, false);

ctx.fill();

}

}

// 이렇게도 할 수 있는데 구문이 너무 길어짐 ,,,,,

// const CircleOne = new Circle(100, 100, 50, "red");

// const CircleTwo = new Circle(200, 200, 20, "darkblue");

// const CircleThree = new Circle(350, 30, 40, "yellow");

// const CircleFour = new Circle(500, 270, 80, "darkgreen");

// const CircleFive = new Circle(550, 80, 50, "pink");

// CircleOne.draw();

// CircleTwo.draw();

// CircleThree.draw();

// CircleFour.draw();

// CircleFive.draw();

const objs = [];

for (let i = 0; i < 20; i++) {

const radius = Math.floor((Math.random() \* 50)) + 10;

const x = Math.random() \* (canvas.width - radius \* 2) + radius;

const y = Math.random() \* (canvas.height - radius \* 2) + radius;

const color = `rgb(${Math.random() \* 255}, ${Math.random() \* 255}, ${Math.random() \* 255})`

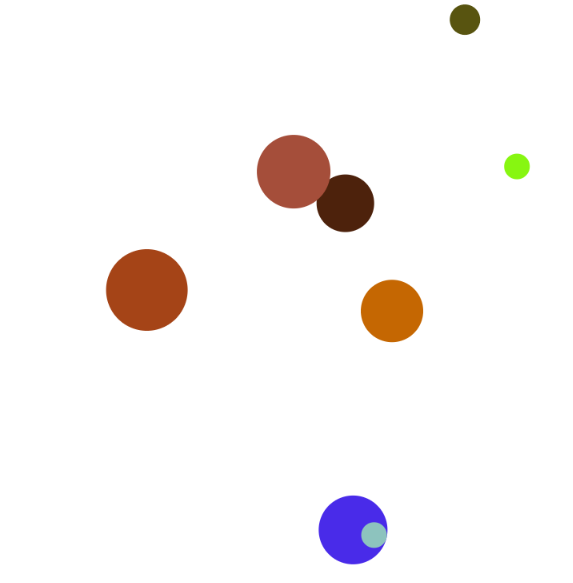
objs.push(new Circle(x, y, radius, color)); //push가 뭐였지 ..

}

for (let i = 0; i < objs.length; i++) {

objs[i].draw();

}



움직이지는 않고 이렇게만 멈춰서 나옴!

에니미 활용해보기

* Script 주소 불러와야함!  
  **<script defer src="https://cdnjs.cloudflare.com/ajax/libs/animejs/3.2.1/anime.js"></script>**

anime({

targets : "ul.menu li",

translateX : 250,

easing : "linear",

loop : true,

direction : "alternate", // 다시 돌아오도록하는 에니메이션

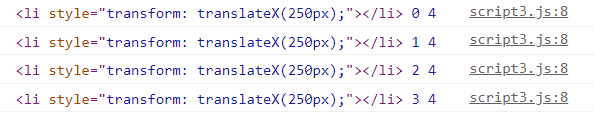
delay : function(el, i, l) {

console.log(el, i, l);

}

})

콘솔창 확인



에니메이션 속도 > 사이트

https://easings.net