9-1 p5.js

p5.js 설치하기 및 캔버스 정의하기

p5.js를 활용하기 위해 다음과 같이 **호스팅된 p5.js 라이브러리를 CDN 방식**으로 불러와주겠습니다. 그리고 앞으로 그림과 애니메이션을 을 그리게 될 캔버스를 셋팅하는 함수 setup()을 정의하여 주겠습니다.

https://codepen.io/AI_R_IM/pen/NWavJWR?editors=1000

```
<body>
  <!-- https://p5js.org/ko/get-started/#settingUp -->
  <!-- src="https://cdn.jsdelivr.net/npm/p5@[p5_version]/lib/p5.js">
  에서 [p5_version] 부분에는 버전 정보(ex 1.3.0)를 입력하여 주세요. -->
  <script src="https://cdn.jsdelivr.net/npm/p5@1.4.0/lib/p5.js"></script>
  <script>
    // 그림과 애니메이션을 그리게 될 캔버스 setup 함수 정의
    function setup() {
      createCanvas(400, 400); // 캔버스의 폭과 높이
      background("orange"); // 캔버스의 배경색 설정
    }
  </script>
</body>
```

캔버스 정의

결괏값

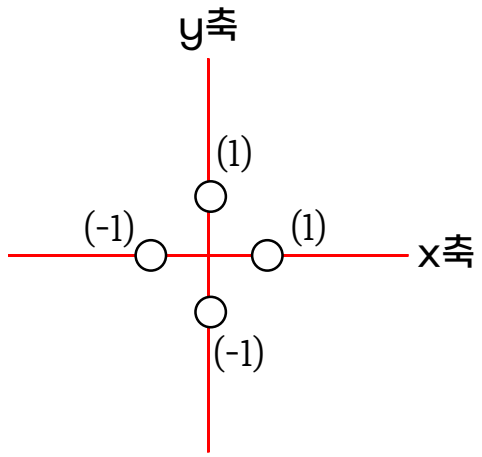
https://p5js.org/ko/get-started/#settingUp

9-1 p5.js

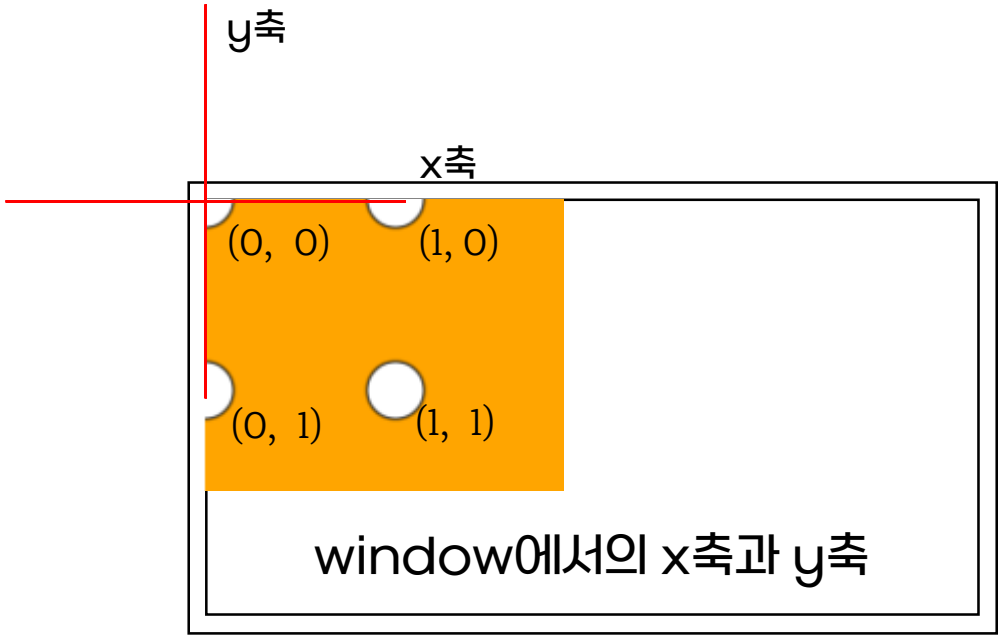
좌표값의 확인 및 p5.js 를 통해 원을 그리기

화면상에 그림을 그리기 위해서는 **'어디에?'** 라는 질문이 따라옵니다. 즉 그림을 그리기 위해서는 **화면상의 좌표**를 정확하게 알고 있어야 한다는 것이죠. 일반적으로 **x축은 오른쪽으로 갈 수록 값이 커지고, 왼쪽으로 갈 수록 값이 작아집니다. y축은 위로갈수록 값이 커지고 아래쪽으로 내려갈 수록 값이 작아집니다.**

그러나 **화면상에서의 y축**은 그림에서 처럼 **아래로 내려갈 수록 값이 커지며, 위로 올라갈 수록 값이 작아집니다.**



일반적인 x축과 y축



window에서의 x축과 y축

9-1 p5.js

좌표값의 확인 및 p5.js 를 통해 원을 그리기

1. p5.js 홈페이지에서 '레퍼런스'를 찾아 접속
https://p5js.org/ko/reference/

p5.js

홈
에디터
다운로드
후원하기
시작하기
레퍼런스
라이브러리

레퍼런스 API 검색

찾는 항목이 없다면, 다음의 페이지를 살펴보세요: p5.sound.
오프라인 버전 다운로드

Color	Environment	Image	Shape
Constants	Events	Lights, Camera	Structure
DOM	Foundation	Math	Transform
Data	IO	Rendering	Typography

도형

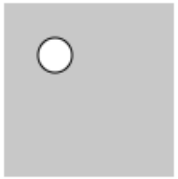
2D 기초 조형	도형 속성	곡선
arc()	ellipseMode()	bezier()
ellipse()	noSmooth()	bezierDetail()
circle()	rectMode()	bezierPoint()
line()	smooth()	bezierTangent()
point()	strokeCap()	curve()
quad()	strokeJoin()	curveDetail()
rect()	strokeWeight()	curveTightness()
square()		curvePoint()
triangle()		curveTangent()

2. 도형에서 '원' 을 의미하는 circle()을 찾아 접속
https://p5js.org/ko/reference/#/p5/circle

레퍼런스 API 검색

circle()

예제



edit reset copy

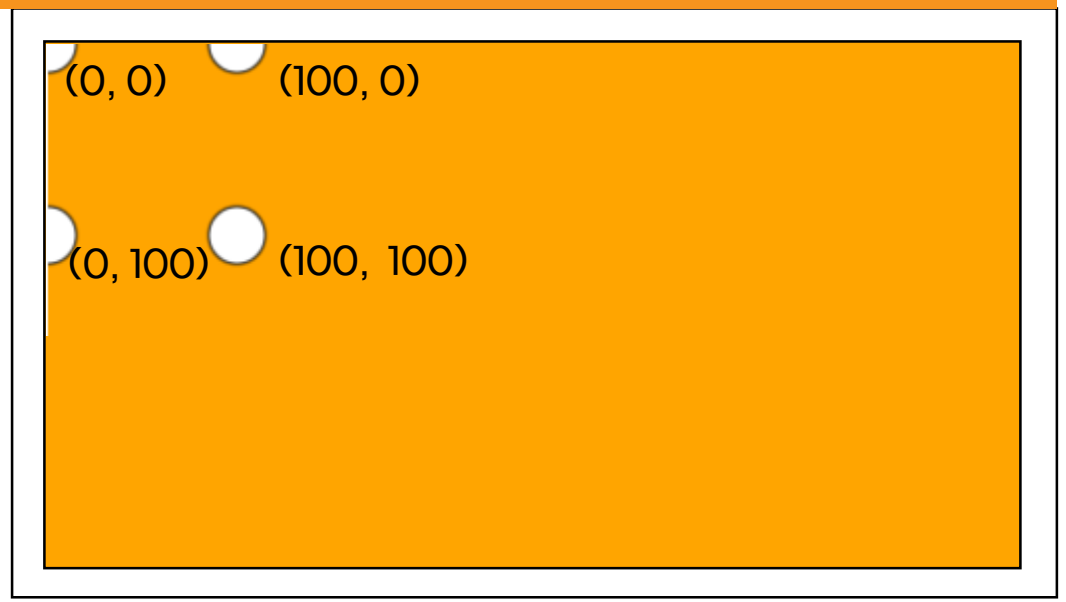
```
// Draw a circle at location (30, 30) with a diameter of
20.
circle(30, 30, 20);
```

9-1 p5.js

좌표값의 확인 및 p5.js 를 통해 원을 그리기

https://codepen.io/AI_R_IM/pen/YzrxRxP?editors=1000

```
<script>
    // 그림과 애니메이션을 그리게 될 캔버스 setup 함수 정의
    function setup() {
        createCanvas(1200,800); // 캔버스의 폭과 높이
        background('orange')    // 캔버스의 배경색 설정
        circle(0,0,30);         // circle(x좌표,y좌표,원의 크기)
        circle(100,0,30);
        circle(0,100,30);
        circle(100,100,30);
    }
</script>
```



window

9-1 p5.js

좌표값의 확인 및 p5.js 를 통해 도형을 그리고 색과 선을 제어하기

도형을 그리는 방식은 원하는 도형의 **레퍼런스를 검색**하여 예제를 확인하고 그려줍니다. 예를들어 **사각형(rect)의 경우** 다음과 같습니다. 참고로 도형의 배경색, 도형의 선 색상 및 두께를 미리 정의하고 도형을 정의하여 줍니다.

1. p5.js 홈페이지에서 'rect'를 검색하여 접속
https://p5js.org/ko/reference/#/p5/rect

레퍼런스 API 검색

원하는 도형을 검색

rect()

예제

edit reset copy

```
// Draw a rectangle at location (30, 20) with a width and
height of 55.
rect(30, 20, 55, 55);
```

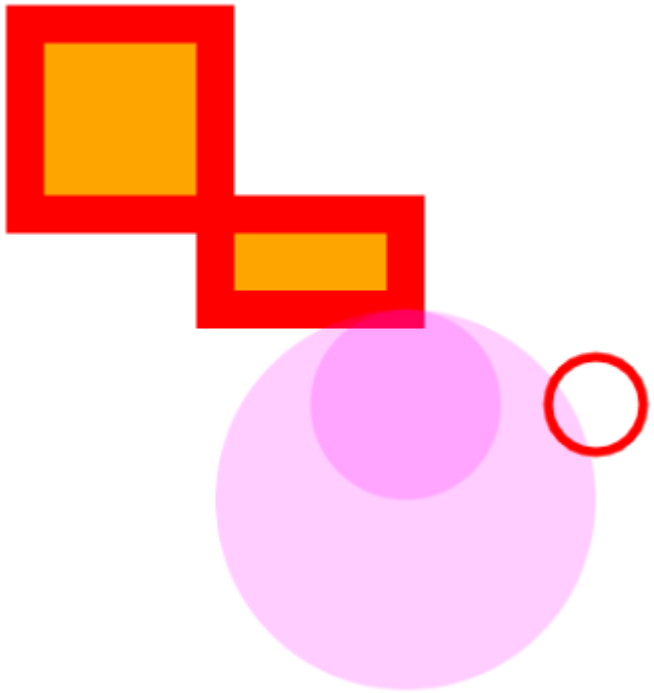
edit reset copy

```
// Draw a rectangle with rounded corners, each having a
radius of 20.
rect(30, 20, 55, 55, 20);
```

edit reset copy

```
// Draw a rectangle with rounded corners having the
following radii:
// top-left = 20, top-right = 15, bottom-right = 10,
bottom-left = 5.
rect(30, 20, 55, 55, 20, 15, 10, 5);
```

2. 도형의 색, 도형의 선을 칠해주는 레퍼런스를 찾아줍니다.
도형의 색 – fill() / 선의 색 – stroke() / 선 두께 – strokeWeight()
https://codepen.io/AI_R_IM/pen/vYeJPVG?editors=1000



9-1 p5.js

p5 를 활용하여 애니메이션 구현하기 1

애니메이션을 구현하기 위해 다음과 같이 draw()를 정의하여야 합니다. **draw()안에 구현된 코드는 빠르게 반복적으로 동작**하게 됩니다. 그러므로 **하나의 원을 그리는 코드는 캔버스에 같은 위치에 같은 크기에 원을 반복적으로 그리는 것**이기에 애니메이션처럼 보이지는 않습니다. **하지만 random()를 활용하여 매번 다른값을 x축, y축, 크기에 준다면 마치 애니메이션처럼** 보여집니다.

https://codepen.io/AI_R_IM/pen/RwLZOaV?editors=1000

```
<script>
    // 그림과 애니메이션을 그리게 될 캔버스 setup 함수 정의
    function setup() {
        createCanvas(1200,800);  // 캔버스의 폭과 높이
        background('white');     // 캔버스의 배경색 설정
    }
    // 처음 setup이 실행된 이후, 그림을 그려주기 위해 빠르게 동작하는 함수
    function draw() {
        // console.log(random(300)); // random() 를 사용해 0~300 사이의 값을 반환
        // circle(200, 200, 100); // 하나의 원을 반복적으로(draw()) 그려줍니다.
        // 원을 그리를 과정에서 다양한 크기를 갖는 원을 반복적으로 그리기 위해 random()를 활용.
        fill('pink')
        stroke('pink')
        rect(random(500), random(500), random(100));

        fill(76,1,32)
        stroke(76,1,32)
        rect(random(500), random(500), random(100));

    }
</script>
```

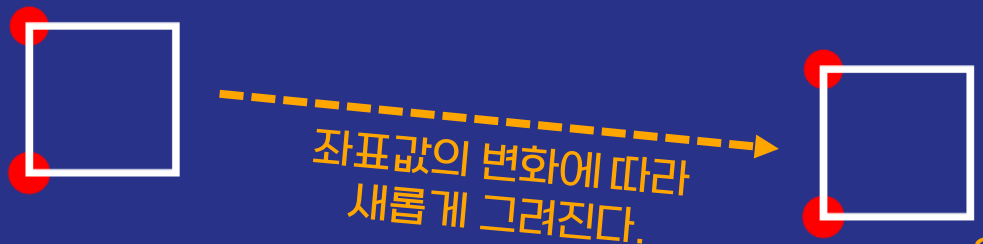
빠르게 동작하는 내부 함수들

9-1 p5.js

p5 를 활용하여 애니메이션 구현하기 2

기존 setup에 위치했던 **background() 함수를 draw() 함수 안에 배치**시키게 되면 **배경이 매번 새롭게** 그려지게 됩니다. 그렇게 되면 draw() 함수에서 생성되는 도형들이 겹치면서 화면을 가리는 것이 아닌, 움직임이는 객체처럼 보일 수 있게 됩니다.

https://codepen.io/AI_R_IM/pen/mdBMZmJ?editors=1010



일정 범위에 도달하게 되면 초기화 혹은
특정 위치로 이동하게 조건문을 설정할 수 있습니다.

9-1 p5.js

p5.js 상호작용

마우스의 움직임을 매개변수로 받아 **원을 그려보는 예제**입니다. 이벤트 인식에서 마우스의 움직임을 받는 **mouseX, mouseY 레퍼런스를 확인**하여 봅니다.

https://codepen.io/AI_R_IM/pen/NWaaPBE

이벤트 인식

가속도	키보드	마우스
deviceOrientation	keyIsPressed	movedX
accelerationX	key	movedY
accelerationY	keyCode	mouseX
accelerationZ	keyPressed()	mouseY
pAccelerationX	keyReleased()	pmouseX
pAccelerationY	keyTyped()	pmouseY
pAccelerationZ	keyIsDown()	winMouseX
rotationX		winMouseY
rotationY		pwinMouseX
rotationZ		pwinMouseY
pRotationX		mouseButton
pRotationY		mouseIsPressed
pRotationZ		mouseMoved()
turnAxis		mouseDragged()
setMoveThreshold()		mousePressed()
setShakeThreshold()		mouseReleased()
deviceMoved()		mouseClicked()
deviceTurned()		doubleClicked()
deviceShaken()		mouseWheel()
		requestPointerLock()
		exitPointerLock()

```
9   <body>
10    <script src="https://cdn.jsdelivr.net/npm/p5@1.4.0/lib/p5.js"></script>
11    <script>
12        function setup() {
13            background(40, 50, 137);
14            createCanvas(windowWidth, windowHeight);
15        }
16        function draw(){
17            // mouseX : 마우스의 현재 수평 위치
18            // mouseY : 마우스의 현재 수직 위치
19            // 반복적으로 '원'을 그립니다. 이때 마우스의 움직임을
20            // 인자로 받습니다.
21            //circle(mouseX, mouseY, random(100));
22            circle(mouseX + random(-10,10), mouseY + random(-10,10), random(100));
23        }
24    </script>
25  </body>
26 </html>
```

9-1 p5.js

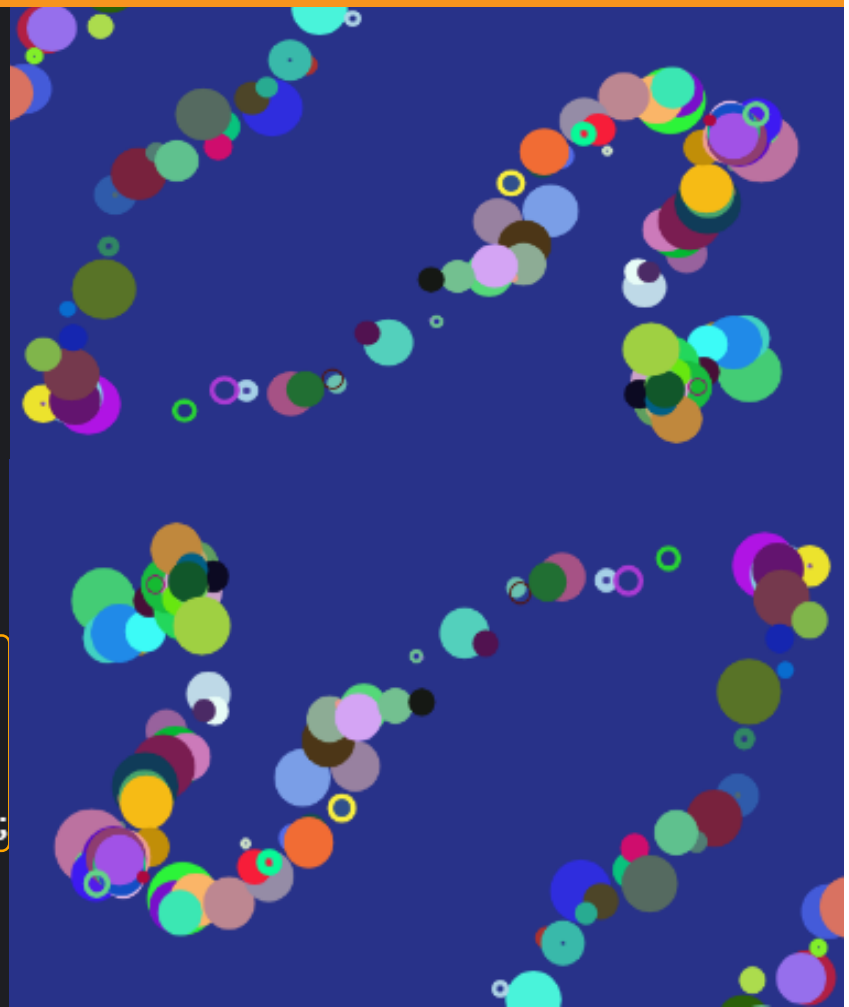
p5.js 상호작용

mouseIsPressed는 **불리언 자료형을 반환**하여 줍니다. 이점을 활용하여 **조건문을 통해 클릭하는 상황인 경우** 다음과 같은 행동을 수행하게 코드를 작성할 수 있습니다.

https://codepen.io/AI_R_IM/pen/KKXXpKP?editors=1010

```
<script>
    function setup() {
        createCanvas(windowWidth, windowHeight);
        background(40, 50, 137);
    }
    function draw(){
        // mouseIsPressed의 반환값은 True와 False 입니다.
        console.log('mouseIsPressed : ',mouseIsPressed);

        // mouseIsPressed의 값이 참(True)라면(클릭 이벤트가 발생하면)
        if (mouseIsPressed){
            // 무작위 원을 그려내는 방식을 취해주도록 합니다.
            fill(random(0,255), random(0,255), random(0,255), random(0,100));
            stroke(random(0,255), random(0,255), random(0,255));
            strokeWeight(random(0,20));
            circle(mouseX + random(-10,10), mouseY + random(-10,10), random(10));
        }
    }
</script>
```

조건문을 추가하여 마우스 클릭 이벤트에 따른 기능을 정의할 수 있습니다.



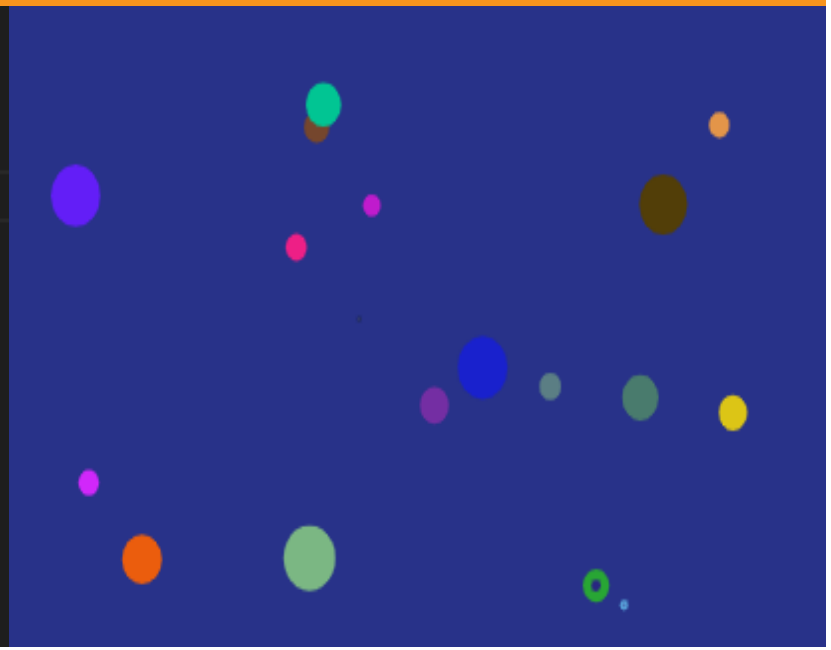
9-1 p5.js

p5.js 상호작용

mousePressed() 함수는 **마우스 클릭 이벤트가 발생**한 경우 **한번의 동작을 수행**합니다.

https://codepen.io/AI_R_IM/pen/OJxxVZK

```
function setup() {
    createCanvas(windowWidth, windowHeight);
    background(40, 50, 137);
}
// mousePressed()는 마우스 클릭이 한번 발생하였을 때 동작하는 함수
function mousePressed() {
    // 마우스가 클릭되면 다음과 같은 도형을 그려줍니다.
    fill(random(0,255), random(0,255), random(0,255), random(0,100));
    stroke(random(0,255), random(0,255), random(0,255));
    strokeWeight(random(0,20));
    circle(mouseX + random(-10,10), mouseY + random(-10,10), random(10));
}
```



9-1 p5.js

p5.js 상호작용

mouseIsPressed와 line()을 활용하여 그림판을 만들어 보겠습니다. **line()는 이전 x,y 좌표와 현재 x,y 좌표를 인자**로 받습니다. draw() 안에 있는 코드들은 무수한 반복을 수행하기에, **line()가 짧은 호흡으로 무한하게 동작**하게 됩니다. 그럼 선은 짧게 그려지지만 이것들이 계속해서 이어지기에 **다양한 선을 출력**해 낼 수 있습니다.

https://codepen.io/AI_R_IM/pen/bGooVNp

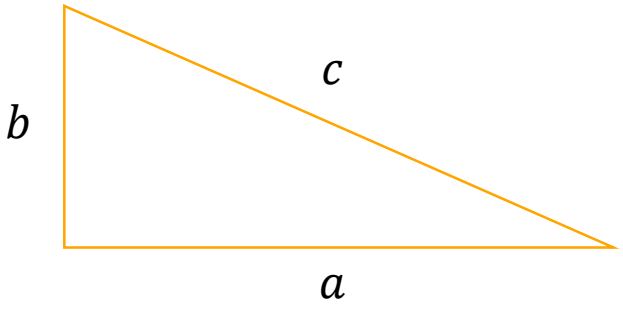
```
<script>
    function setup() {
        createCanvas(windowWidth, windowHeight);
        background(40, 50, 137);
    }
    // mouseIsPressed를 사용해 마우스 클릭이 발생하면 line()를 통해
    // 선을 그리도록 합니다.
    function draw() {
        if(mouseIsPressed){
          //line(이전 마우스X, 이전 마우스Y, 현재 마우스X, 현재 마우스Y)
            line(pmouseX, pmouseY, mouseX, mouseY);
        }
    }
</script>
```

이전 마우스 포인트를 가져옵니다.

9-1 p5.js

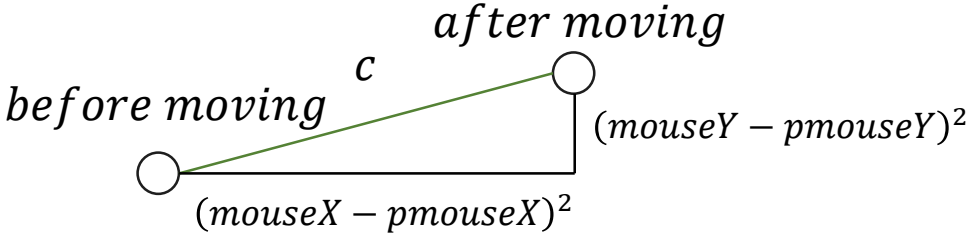
p5.js 상호작용

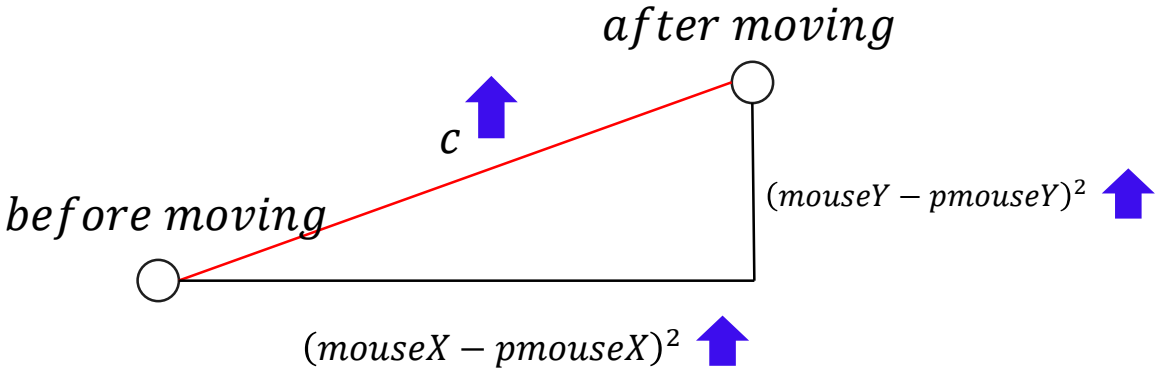
마우스의 **움직임 속도를 고려하여 커서의 크기가 달라지는 예제**입니다. 속도를 직접적으로 구하는 공식이 아닌 **피타고라스의 정리를 활용**하고자 합니다. 피타고라스의 정리는 다음과 같습니다.

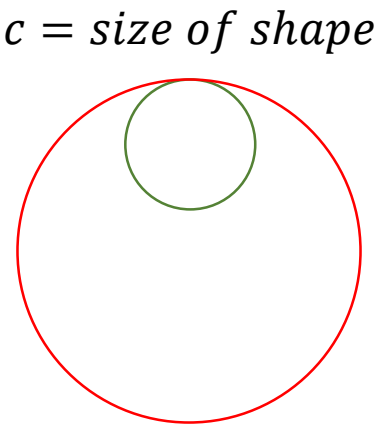


피타고라스의 정리

$$a^2 + b^2 = c^2$$







우린 마우스가 움직이면 **총 네 가지 값**을 얻습니다. **x축의 현재 위치와 이동한 위치, y축의 현재 위치와 이동한 위치**. 즉, 피타고라스의 정리에서 필요한 **a의 값과 b의 값**을 구할 수 가 있으며, 동시에 c의 값 또한 구할 수가 있게 되는 것이죠.
여기서 마우스가 빠르게 움직인다면 a의 값과 b의 값은 커질 수 밖에 없습니다. **동시에 c의 값 또한 커지게 됩니다.** 여기서 c의 값을 도형의 크기를 제어하는 매개변수에 할당하게 된다면, **마우스 움직임이 빨라질 수록 도형의 크기 또한 자연스럽게 커지는 효과**를 얻게 되는 것입니다.

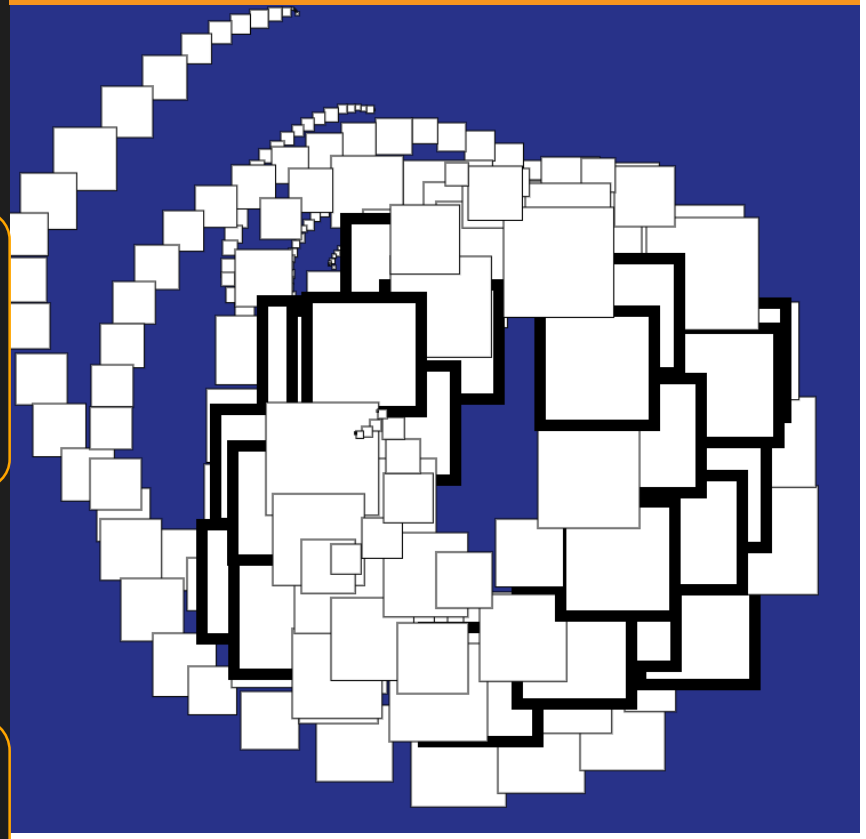
9-1 p5.js

```
<script>
    function setup() {
        createCanvas(windowWidth, windowHeight);
        background(40,50,137);
    }

    function draw(){
        if (mouseIsPressed){
            // 피타고라스의 정리를 활용하기 위해 삼각형의 x축과 y축을 변수화 합니다.
            pa = pow(mouseX - pmouseX, 2);
            pb = pow(mouseY - pmouseY, 2);
            // 빗변을 구하는 공식 c^2 = a^2 + b^2
            pc_speed = sqrt(pa + pb)

            // 마우스가 이동하는 거리가 길어지면 빗변의 길이도 함께 증가
            // 더 큰 도형을 그릴 수 가 있게 됩니다.
            // 만약 움직임의 속도가 너무 빠르면 빗변의 길이가 너무 길어지기에
            // 조건을 걸어 빗변의 길이를 제한하여 줍니다.
            if (pc_speed <= 100){
                strokeWeight(1);
                rect(mouseX, mouseY, pc_speed)
            } else {
                strokeWeight(10);
                rect(mouseX, mouseY, 100)
            }
    }
}
</script>
```

피타고라스의 정리

빗변의 길이 증가 제한

https://codepen.io/AI_R_IM/pen/OJxxjdw?editors=1010



9-1 p5.js

p5.js 를 활용한 이미지 불러오기

loadImage()를 통해 **자신의 로컬 경로에 위치한 이미지 파일**을 불러올 수 있습니다. 그리고 draw() 함수 내부에 **image()를 선언하여 화면상에 출력할 이미지 크기를 제어**할 수 있습니다.

https://codepen.io/AI_R_IM/pen/dyVVEGV?editors=1011

```
<script>
  let img;                                    // 이미지를 담을 변수 선언
  function setup() {
    createCanvas(windowWidth,windowHeight);  // 화면의 크기 지정
    img = loadImage('https://placekitten.com/400/250'); // 이미지 변수화
  }

  function draw() {
    // 이미지(img)를 화면 좌표 (0,0) 위치에 기존 크기로 보이게 합니다.
    image(img, 0,0,img.width, img.height);
  }
</script>
```

9-1 p5.js

p5.js 마우스를 천천히 따라오는 객체 - 1

```
<script>
  function setup() {
    createCanvas(720, 400); // 캔버스의 크기 지정
    noStroke();             // 윤곽선 해제
  }

  function draw() {
    background(247, 148, 30); // 배경화면 색상 지정
    textAlign(RIGHT);         // 텍스트 정렬
    textSize(15)              // 폰트 사이즈 설정
    drawWords(mouseX, 50, 50);  // 마우스 움직임을 바로 출력하도록 drawWords()함수 정의
    drawWords(mouseY, 100, 50); // drawWords(객체, 글자출력 X, 글자출력Y)

    let targetX = mouseX;     // 마우스의 x,y 축 값을 targetX,Y로 변수화
    let targetY = mouseY;

    fill(255);
    ellipse(targetX, targetY, 75, 75);  // 원을 그리는 예제
                              // (x, y, w, h)
  }
```

마우스를 따라오는 원 생성

```
  function drawWords(input_text, x, y) {
    fill(0);                  // 글자 색상
    text(input_text, x, y);   // 텍스트가 그려질 위치(x,y)
  }
</script>
```

화면에 글자 출력 함수

https://codepen.io/AI_R_IM/pen/bGooRog?editors=1001

p5.js의 draw()에 정의된 코드는 **무한하게 반복하는 성질**을 가지고 있습니다. 이를 활용하여 **마우스 커서를 천천히 따라오는 객체를 생성**하여 보겠습니다.

9-1 p5.js

p5.js 마우스를 천천히 따라오는 객체 - 2

```
function draw() {
  background(247, 148, 30); // 배경화면 색상 지정
  textAlign(RIGHT);         // 텍스트 정렬
  textSize(15)              // 폰트 사이즈 설정
  drawWords(mouseX, 50, 50);  // 마우스 움직임을 바로 출력하도록 drawWords()함수 정의
  drawWords(mouseY, 100, 50); // drawWords(객체, 글자출력 X, 글자출력Y)

  let targetX = mouseX;     // 마우스의 x,y 축 값을 targetX,Y로 변수화
  let targetY = mouseY;

  // 원 그리는 과정에서 의도적 딜레이 발생시키기
  let dx = targetX - x;     // 현재 마우스의 x축 값에서 1을 빼줍니다.
  x += dx * easing;         // 그 값에 딜레이 수치를 곱해주고 그 값을 다시 x에 담아줍니다.
                            // 이 과정을 반복하게 되면 x값은 서서히 targetX 값이 됩니다.

  let dy = targetY - y;     // 같은 방식의 y값 산출
  y += dy * easing;

  drawWords(x.toFixed(2), 50, 100);   // x,y값을 화면상에 출력합니다.
  drawWords(y.toFixed(2), 100, 100);

  fill(255);
  ellipse(x, y, 75, 75);  // 원을 그리는 예제
```

https://codepen.io/AI_R_IM/pen/OJxOoGQ

draw()의 무한 반복 동작 성질을 이용합니다.

원을 그리기 위해 필요한 x축의 값과 y축의 값에
천천히 접근하도록 하는 수식을
코드화 하여 줍니다.

9-1 p5.js

p5.js 웹캠 제어

p5.js를 활용하여 **웹캠을 활용**해 보도록 하겠습니다. 다음과 같이 **createCapture() 함수에 VIDEO 타입 이라 설정**하고 화면의 크기를 설정하는 형식으로

https://codepen.io/AI_R_IM/pen/YzrEJyQ

```
<script>
  // 화면에 보여지는 장면을 담아줄 변수 생성
  let capture;

  function setup() {
    createCanvas(800, 600);          // 캔버스의 크기를 설정합니다.
    capture = createCapture(VIDEO); // VIDEO 타입으로 설정합니다.
    capture.size(800, 600);          // 캔버스와 동일하게 화면크기를 제어합니다.
    capture.hide();                  // 하단의 draw() 함수에 image()로 화면이 2개가 출력되기에
                                     // hide()로 하나를 감추도록 합니다.
  }

  function draw() {
    background(255);
    image(capture, 0, 0, 800, 600); // 화면 크기 설정

    // 다양한 필터 적용 가능
    // filter(THRESHOLD,0.5);
    // filter(INVERT)
    // filter(BLUR, 3)
  }

</script>
```

9-1 p5.js

p5.js 를 활용한 이미지 분류-1

p5.js를 활용하여 이미지 분류를 수행하여 보겠습니다. 가장 먼저 https://ml5js.org/ 에 접속하여 CDN 방식으로 p5.js를 비롯한 ml5를 불러와주도록 하겠습니다.

https://codepen.io/AI_R_IM/pen/BawmqZq?editors=1001

CDN 방식으로 p5.js 및 ml5 불러오기

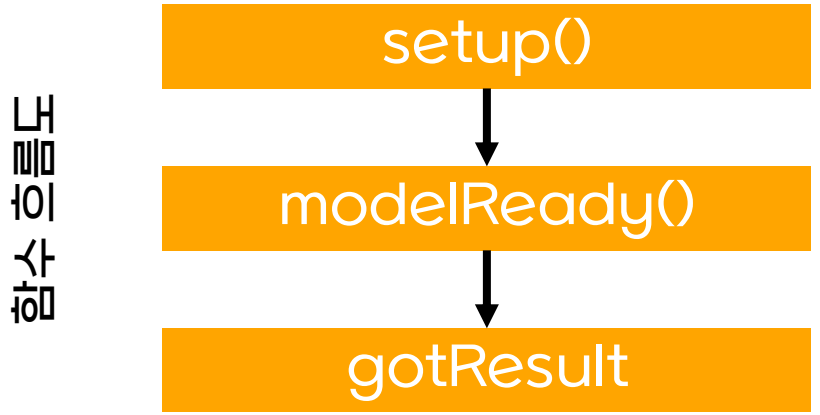
```
3   <head>
4     <meta charset="UTF-8" />
5     <meta name="viewport" content="width=device-width, initial-scale=1.0" />
6     <meta http-equiv="X-UA-Compatible" content="ie=edge" />
7     <title>P5.js Sketch</title>
8     <!-- CDN 방식으로 p5.js 를 불러옵니다.   -->
9     <script src="https://cdnjs.cloudflare.com/ajax/libs/p5.js/1.1.9/p5.min.js"></script>
10    <script src="https://cdnjs.cloudflare.com/ajax/libs/p5.js/1.1.9/addons/p5.dom.min.js"></script>
11    <!-- https://learn.ml5js.org/#/ 접속하여 ml5를 불러오는 방법을 확인합니다. -->
12    <script src="https://unpkg.com/ml5@latest/dist/ml5.min.js"></script>
13  </head>
```

분류할 이미지, 모델, 모델 이름 변수화

```
14  <body>
15    <script>
16
17      // 이미지, 로드된 모델, 사용할 모델 이름을 변수화 합니다.
18      let img, classifier;
19      let load_model = 'MobileNet';
```

```
// 화면 크기와 이미지 분류에 사용할 이미지를 준비합니다.
function setup() {
  createCanvas(400, 400);
  img = loadImage('/CODE/images/cat.png');

  // ml5.imageClassifier(model, callback)은 분류를 위한 모델 호출 함수
  // 'MobileNet' 이 가장 보편적으로 사용
  // 티처블 머신으로 생성한 모델도 사용 가능합니다.
  classifier = ml5.imageClassifier(load_model, modelReady);
}

// modelReady()는 분류 모델(classifier)을 가지고 분류를 수행
function modelReady(){
  // 모델을 불러와 분류를 수행하는 함수입니다.
  // 정상적으로 동작하면 안내 문구를 콘솔창에 띄워줍니다.
  console.log(`${load_model} 모델 불러오기 완료`);
  // classifier에 접근해 분류를 수행(classify)합니다.
  // 그리고 모델이 정상적으로 동작하지 않는 경우를 대비해 콜백함수를 심어둡니다.
  // gotResult()는 결괏값을 송출하는 함수입니다.
  classifier.classify(img, gotResult);
}
// 결과를 콘솔창으로 출력합니다
function gotResult(err, results){
  // 만약 err값이 true라면 에러를 그대로 송출합니다.
  if(err){
    console.error(err);
    return;
  }
  // 에러가 없다면 정상적으로 분류 결과를 콘솔창에 출력.
  console.log('----------결과----------');
  console.log(results);
}
```

9-1 p5.js



콘솔창

```
community.js:4
🌈🌈🌈🌈🌈🌈🌈🌈🌈🌈🌈🌈🌈🌈🌈🌈🌈🌈🌈
🌟 Thank you for using ml5.js v0.7.1 🌟

Please read our community statement to ensure
that the use of this software reflects the values
of the ml5.js community:
↳ https://ml5js.org/about

Reporting:
↳ https://github.com/ml5js/ml5-library/issues
↳ Email: info@ml5js.org
🌈🌈🌈🌈🌈🌈🌈🌈🌈🌈🌈🌈🌈🌈🌈🌈🌈🌈🌈
MobileNet 모델 불러오기 완료                    1111.html:36
----------결과----------  ▼결과값▼              1111.html:51
                                                1111.html:52
▼(3) [{…}, {…}, {…}]
  ▶0: {label: 'teddy, teddy bear', confidence: 0.6765099763870239}
  ▶1: {label: 'mask', confidence: 0.061180680990219116}
  ▶2: {label: 'coffee mug', confidence: 0.015446235425770283}
    length: 3
  ▶[[Prototype]]: Array(0)
```

9-1 p5.js

p5.js 를 활용한 이미지 분류-2

앞선 예제는 분류 대상 이미지와 분류 결괏값을 콘솔창에 표시하였습니다. 이번에는 **콘솔창이 아닌 화면에 출력하는 과정**을 함께 살펴보겠습니다.

https://codepen.io/AI_R_IM/pen/MWEOPzK?editors=1000

프로그램 실행 순서

setup()
캔버스 화면 크기 지정 및 이미지 출력

preload()
모델 및 이미지 불러오기

↓

modelReady()
분류 수행

↓

gotResult()
결괏값 화면 출력

이미지 →

결과값 →

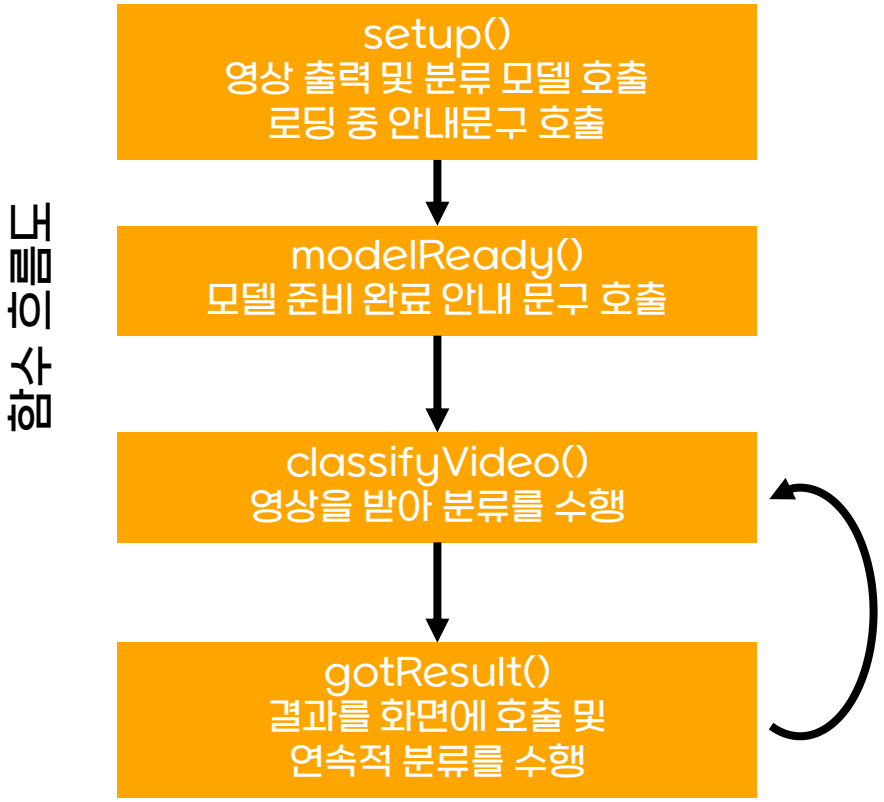
teddy, teddy bear
67.651%

9-1 p5.js

p5.js 를 활용한 실시간 영상 분류

사전 학습된 모델을 불러와 **웹캠을 통해 받은 영상을 연속적으로 분류**합니다. 그리고 영상과 정확도를 화면에 출력하는 예제입니다.

https://codepen.io/AI_R_IM/pen/QWqOJyj?editors=1010



객체 감지를 위해 인공지능 MobileNet 모델을 불러오고 있습니다.

영상 분석 결과 : person
정확도 : 0.27

9-1 p5.js

p5.js 를 활용한 객체 탐지-1

객체 탐지 모델을 불러오고 결과값을 console창에 출력하는 예제 입니다. 객체 탐지 모델은 'yolo'와 'cocossd'가 존재합니다.

https://codepen.io/AI_R_IM/pen/OJxOdMZ?editors=1111

```
<script>
  let detector;      // 모델을 담을 변수 정의
  let img;           // 이미지를 담는 변수 정의

  // 이미지를 불러오는 함수 정의
  function preload(){
    img = loadImage('/CODE/images/cat.png');
  }

  function setup() {
    createCanvas(img.width, img.height);              // 이미지 크기만큼의 화면 생성
    detector = ml5.objectDetector('YOLO', modelReady); // 'yolo' 모델을 불러오도록 합니다.
  }

  function modelReady() {                             // 모델 준비 완료를 알려주는 함수 정의
    console.log('인공지능 모델 준비 완료.');
    detector.detect(img, gotResult);                  // 이미지를 넣어 객체 탐지 수행
  }

  function gotResult(err, results){                   // 결괏값 출력 함수 정의
    if(err){                                          // 에러가 있는 경우
      console.error(err);
      return;
    }
    console.log(results);                             // 없는경우 결과 출력(콘솔)
  }
</script>
```

community.js:4

Thank you for using ml5.js v0.7.1

Please read our community statement to ensure
that the use of this software reflects the values
of the ml5.js community:
↳ https://ml5js.org/about

Reporting:
↳ https://github.com/ml5js/ml5-library/issues
↳ Email: info@ml5js.org

인공지능 모델 준비 완료. 1111.html:30

1111.html:39

```
▼ [{…}]
  ▼ 0:
      confidence: 0.8909937739372253
      height: 329.4263610839844
      label: "teddy bear"
    ▶ normalized: {x: 0.16495317679185134, y: 0.02585…
      width: 284.5906295776367
      x: 68.62052154541016
      y: 10.7576904296875
    ▶ [[Prototype]]: Object
    length: 1
  ▶ [[Prototype]]: Array(0)
```

정확도

이미지 내의 객체 좌표값

9-1 p5.js

p5.js 를 활용한 객체 탐지-2

이번에는 객체 탐지에 사용되는 **이미지와 결과값을 화면에 출력**하도록 하겠습니다. (※ 정확도를 위해 'cocossd' 모델을 사용하도록 하겠습니다.)

https://codepen.io/AI_R_IM/pen/KKXyJaL?editors=1000

preload()
이미지를 변수화 하는 함수

setup()
캔버스를 정의하고 모델을 변수

modelReady()
이미지를 모델에 넣어 객체 탐지

gotResult()
결과를 화면에 호출 및 연속적 분류를 수행

```
// 이미지를 담는 함수 정의
function preload(){
  img = loadImage('/CODE/images/cat.png');
}

// 캔버스를 정의하고 모델을 변수화 하여 줍니다.
function setup() {
  createCanvas(img.width, img.height);
  detector = ml5.objectDetector(load_model, modelReady);
}

// 이미지를 모델에 넣어 객체 탐지.
function modelReady() {
  console.log(`AI ${load_model} 준비 완료.`);
  detector.detect(img, gotResult);
}
```

```
// 결과 반환 함수 정의
function gotResult(err, results){
  if(err){
    console.error(err);
    return;
  }
  image(img,0,0);        // 이미지를 (0,0) 위치에 출력
  objects = results;     // 결괏값(results)을 변수 objects에 할당
  // 객체를 탐지하는 사각형 정의
  noFill();              // 사각형 채우기 제거
  stroke(247, 148, 30); // 테두리 색 정의
  strokeWeight(1);       // 테두리 두께 정의
  // 사각형 위치 및 높이, 너비(x,y,w,h) 할당
  rect(objects[0].x, objects[0].y, objects[0].width, objects[0].height);

  // 객체의 이름 정의
  textSize(16);        // 텍스트 크기
  strokeWeight(5);     // 글자 테두리 두께 정의
  fill(0, 0, 0);       // 색상 정의
  // 객체 이름 출력(text, x, y)
  text(objects[0].label, objects[0].x+5, objects[0].y-5);
}
```

9-1 p5.js

p5.js 를 활용한 객체 탐지-3

이번에는 하나의 객체 탐지가 아닌 **다수의 객체 탐지를 위해 결과를 화면에 출력하여 주는 gotResult()를 조금 수정**하여 줍니다. 객체를 담는 변수 results에 다수의 결괏값이 들어가 있기에 이를 **반복문 인덱스 i로 하여 사각형으로 감싸주고, 글자를 작성**하여 줍니다.

https://codepen.io/AI_R_IM/pen/dyVZaZV?editors=1010

```
function gotResult(err, results){
  if(err){
    console.log(err);
    return;
  }
  image(img,0,0);
  objects = results;
  // 다수의 객체를 탐지하기 위한 반복문 입니다.
  // i<objects.length; 를 통해 탐지된 객체만큼 반복문을 수행
  for(let i=0; i<objects.length; i++){
    noFill();
    stroke(247, 148, 30);
    strokeWeight(1)
    // 탐지된 오브젝트들을 반복문 인덱스 i 통해 사각형 처리 합니다.
    rect(objects[i].x, objects[i].y, objects[i].width, objects[i].height);
    textSize(16);
    strokeWeight(5);
    fill(255, 255, 255);
    // 탐지된 오브젝트들의 이름을 반복문 인덱스 i 통해 출력합니다.
    text(objects[i].label, objects[i].x+5, objects[i].y-5);
  }
}
```

let load_model = 'yolo';

let load_model = 'cocossd';

9-1 p5.js

p5.js 를 활용한 객체 탐지-4

이번에는 **웹캠을 활용하여 노트북에 송출되는 화면에서 객체 탐지**를 수행하여 줍니다.

https://codepen.io/AI_R_IM/pen/QWqOYrX?editors=1111