



Chapter 8

캔버스(Canvas)-1



자바스크립트의 개요



▣ 자바스크립트 (Javascript)

- **넷스케이프(Netscape)와 선 마이크로시스템(Sun Microsystem)에서 개발한 Client-Side Script 언어**
- **HTML 기능을 수용하면서 프로그래밍 개념을 대폭 수용**
 - 클라이언트(브라우저)에서 실행 가능한 언어로 HTML과 함께 사용
 - 클라이언트에서 실행되므로 별도의 번역과정이 필요 없음
- **역동적이고 시각적인 홈페이지를 작성**
 - HTML이나 CSS로 표현할 수 없는 부분 개발 가능
- **인터넷 프로그래밍에 필수적으로 사용됨**
 - 클라이언트 (브라우저)로부터 서버(웹 서버)로의 효율적인 데이터의 전송





▣ Java와 Javascript의 비교

	Java	Javascript
작성방법	별도의 클래스 파일로 작성	HTML 코드 내에 삽입 별도의 파일로 작성 (.js)
언어해석	컴파일러(Compiler)	인터프리터(Interpreter)
실행방식	서버에서 컴파일, 클라이언트에서 수행	클라이언트에서 해석되어 수행
변수 선언	자료형 선언 필요	자료형 선언 불필요
객체지향	Fully Object-Oriented	Partially Object-Oriented
보안성	보안성 우수	보안성 미흡
난이도	어려움	쉬움





▣ 자바스크립트의 특징

- **Client-Side Script Language**
 - 클라이언트에 의해 해석되어 실행됨
 - 컴파일이 필요 없으며, HTML 문서 내에 포함되어 실행됨
- **Object-Oriented Language**
 - 객체, 메소드, 속성을 지원 (클래스와 상속은 지원하지 않음)
- **C언어를 기반으로 프로그래밍 작성**
 - 객체지향의 일부 기능을 지원하나 C언어를 기반으로 프로그래밍 수행
 - C언어의 기본적인 개념만 있으면 배우기 쉬움
- **보안에 취약하며, 제공 메소드의 한계성**





▣ 자바스크립트의 기본 구조

```
<SCRIPT TYPE="text/javascript">
<!--
    자바스크립트 코드 ;
//-->
</SCRIPT>
```

- 자바스크립트 코드는 **<SCRIPT>**와 **</SCRIPT>** 태그 사이에 존재
- 자바스크립트 코드는 일반적으로 **<HEAD>** 태그 내에 존재
 - : **<BODY>** 태그 내에 존재해도 무관하나 통상적으로 **<HEAD>** 태그 내에 작성
- **<SCRIPT>** 태그 내의 HTML 주석 구문은 생략 가능
 - : 주석은 자바스크립트를 지원하지 않은 브라우저에서 소스의 출력을 방지하기 위함
 - : 대부분은 브라우저는 자바스크립트를 지원하므로 주석의 삽입은 의미 없음





■ 내장형

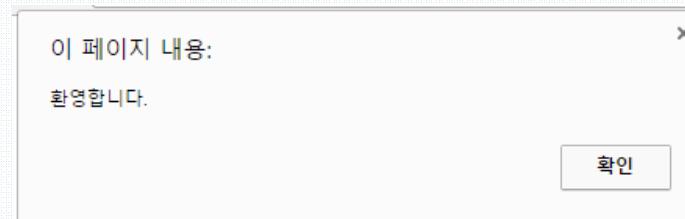
- 문서 내에 기본 구조 형태로 삽입하는 방법
- 일반적으로 <HEAD> 태그 내에 위치함

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">

    <SCRIPT TYPE="text/javascript">
        alert('환영합니다.');
    </SCRIPT>

</head>
<body>

</body>
</html>
```





▣ 행입력형

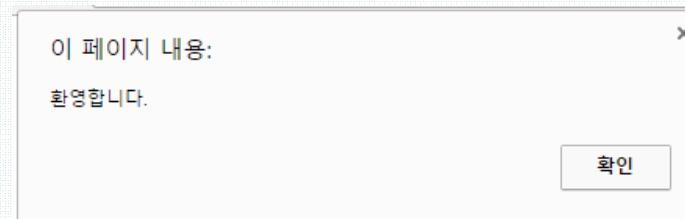
- **HTML 문서의 태그 내에 이벤트 핸들러와 함께 사용하는 방법**
 - 간단한 자바스크립트 코드의 실행을 위해 사용

```
<!DOCTYPE html>
<html>

<head>
    <meta charset="UTF-8">
</head>

<body onLoad="javacsript:alert('환영합니다.')">
</body>

</html>
```





▣ 링크형

- 자바스크립트 코드를 별도의 파일(.js)로 작성하여 링크를 적용해
 - 코드의 보안을 위한 경우 사용

intro.js

```
alert('환영합니다.');
```

```
<!DOCTYPE html>
<html>

<head>
    <meta charset="UTF-8">
    <SCRIPT TYPE="text/javascript" SRC="intro.js"></SCRIPT>
</head>

<body>
</body>

</html>
```





▣ 함수 호출형

- 내장형을 사용해 자바스크립트 함수를 생성하고, 행 입력형을 사용해 자바스크립트 함수를 호출하여 실행하는 방식
 - 프로그래밍에서 주로 사용됨

```
<!DOCTYPE html>
<html>

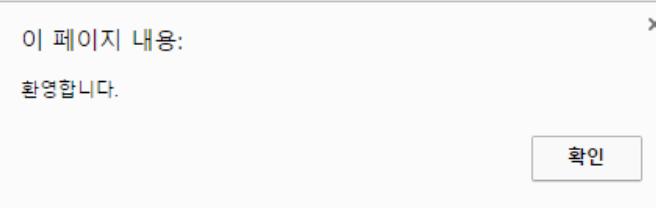
<head>

<SCRIPT TYPE="text/javascript">
    function hello() {
        alert('환영합니다.');
    }
</SCRIPT>

</head>

<body onLoad="javacsript:hello()">
</body>

</html>
```





- ▣ 한 라인에 하나의 실행 문장 사용을 원칙으로 함
 - 한 라인에 두 개 이상의 실행 문장을 작성하고자 할 경우 세미콜론(;)으로 구분
- ▣ 인용부호의 사용
 - 인용부호가 중복되는 경우
 - 외부에는 이중인용부호로, 내부에는 단일인용부호를 사용
 - 인용부호를 문자로 사용할 경우, 문자 앞에 역슬래시를 사용
 - [예] alert("저는 ¢'홍길동¢ 입니다.")
- ▣ 변수의 자료형은 선언할 필요가 없음
 - 입력되는 데이터의 종류에 따라 자동으로 변수형이 결정됨
 - 묵시적인 형 변환 가능
- ▣ 대소문자는 구분됨





캔버스의 개요



▣ 캔버스(Canvas)

- **기존의 HTML에서는 도형적인 요소를 작성하는 방법이 없었음**
 - 직선, 사선, 원 사각형, 타원, 다각형 들의 도형적인 요소를 작성하는 태그가 지원되지 않았음
 - 그림의 경우 요소를 사용해 삽입할 수 있었으나 크기만을 수정할 수 있을 뿐 이미지에 대한 효과를 지정하는 것은 불가능했음
- **캔버스(Canvas)는 브라우저에 도형적인 요소를 작성하기 위해 생성하는 영역을 의미**
 - 브라우저 내에 도형적 요소를 작성하기 위해 설정한 영역으로 각 도형적인 요소들은 설정한 캔버스 내에서만 작성될 수 있음
 - 기본적인 그림이나 도형을 표현하고 효과 지정 가능
 - 캔버스 내에 텍스트를 표현하고 텍스트의 효과 지정 가능
- **캔버스의 생성과 도형요소 작성**
 - 브라우저에 그림을 그리기 위한 캔버스 영역은 <canvas> 요소를 사용해 지정
 - 실제 도형적 요소를 작성하는 것은 자바스크립트와 캔버스를 위한 메서드들을 사용





■ <canvas> 요소

- 캔버스 영역을 생성하기 위한 태그
 - 이름, 너비, 높이, 테두리를 지정

```
<canvas id="캔버스아이디" width="너비" height="높이" [테두리]>  
    텍스트 | 이미지  
</canvas>
```

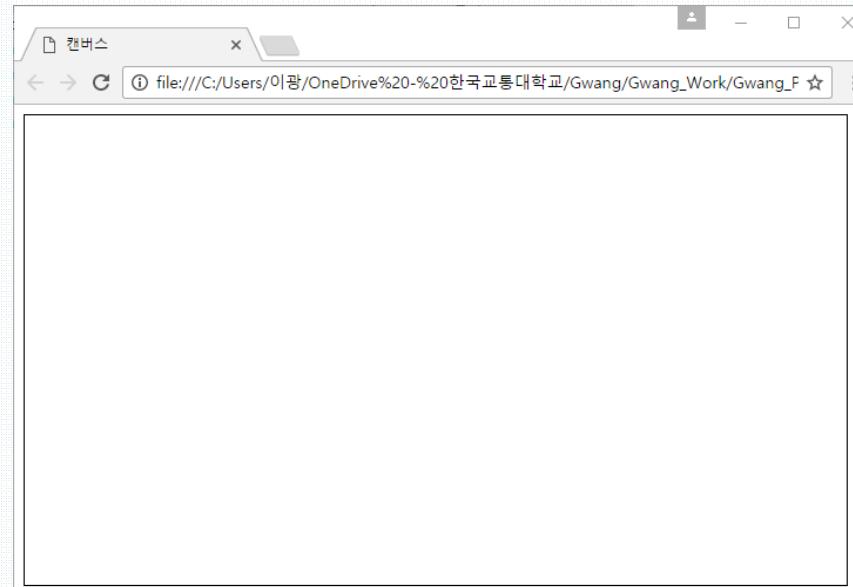
속성 값	설명
id	캔버스의 구분하는 아이디로, 임의 지정 가능(생략 불가능)
width	캔버스의 가로(폭) 크기를 양의 정수(픽셀)로 지정 (생략 불가능) 지정하지 않으면 기본 값으로 300이 지정됨
height	캔버스의 세로(높이) 크기를 양의 정수(픽셀)로 지정(생략 불가능) 지정하지 않으면 기본 값으로 150이 지정됨
테두리	width와 height가 이루는 캔버스의 태두리 속성을 지정 (생략 가능)
텍스트 이미지	캔버스를 지원하지 않는 브라우저에서 캔버스 영역 대신 출력될 텍스트나 이미지 지정





■ <canvas> 요소의 예

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <title> 캔버스 </title>
</head>
<body>
    <canvas id="canvas" width="700" height="400" style="border:solid 1px #000000">
        canvas 사용하기
    </canvas>
</body>
</html>
```





▣ 캔버스 컨텍스트(context) 생성

- 캔버스 내에 도형 요소들의 생성은 캔버스 컨텍스트(canvas context)를 통해서 처리됨
 - 캔버스는 컨텍스트를 위한 컨테이너 역할만을 수행
 - 캔버스에 도형요소나 그림을 생성하려면 반드시 캔버스 컨텍스트를 생성해야 함

```
var 캔버스객체 = document.getElementById("캔버스아이디");
var 컨텍스트 = 캔버스객체.getContext("2d");
```

```
var canvas = document.getElementById("myCanvas");
var context = canvas.getContext("2d");
```

또는

```
var 컨텍스트 = document.getElementById("캔버스 아이디").getContext("2d");
```

```
var context = document.getElementById("myCanvas").getContext("2d");
```



캔버스의 개요



▣ 캔버스 컨텍스트(context) 생성의 예 - 1

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <title> 캔버스 </title>

    <script type="text/javascript">
        function example()
        {
            var canvas = document.getElementById('canvas');
            var context = canvas.getContext('2d');
        }
    </script>
</head>

<body onload="example()">
    <h2>캔버스 컨테스트 생성</h2><hr>

    <canvas id="canvas" width="500" height="500" style="border:solid 2px red;">
        canvas 사용하기
    </canvas>

</body>
</html>
```





▣ 캔버스 컨텍스트(context) 생성의 예 - 2

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <title> 캔버스 </title>
</head>

<body onload="example()">
    <h2>캔버스 컨테스트 생성</h2><hr>

    <canvas id="canvas" width="500" height="500" style="border:solid 2px red;">
        canvas 사용하기
    </canvas>

    <script type="text/javascript">
        function example()
        {
            var canvas = document.getElementById('canvas');
            var context = canvas.getContext('2d');
        }
    </script>

</body>
</html>
```





도형 요소 작성

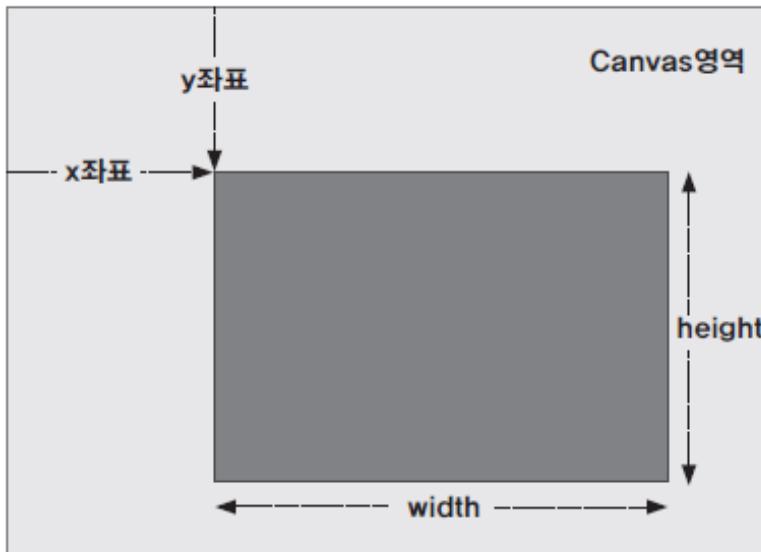


▣ 사각형 작성

- 특정 시작점을 기준으로 너비와 높이를 가진 사각형을 작성
 - 시작점 x좌표, 시작점 y좌표, 너비, 높이를 나타내는 4가지 값이 필요

▣ 사각형 그리기를 위한 메서드

- **fillRect()**
 - 속이 채워진 사각형
- **strokeRect()**
 - 테두리만 있는 사각형
- **clearRect()**
 - 내부 영역을 비우는 사각형





▣ fillRect() 메서드

- 색이 채워진 사각형을 작성하는 메서드
 - 기본 값은 검정색임
 - fillStyle() 메서드를 사용해 내부 색을 변경할 수 있음

```
fillRect( x좌표, y좌표, 너비, 높이 )
```

```
fillRect( 100, 100, 50, 50 )
```

속성 값	설명
x좌표, y좌표	사각형의 왼쪽 상단 모서리의 좌표
너비, 높이	사각형의 너비와 높이 너비나 높이가 0일 경우 사각형은 출력되지 않음





▣ strokRect() 메서드

- **태두리만 있는 사각형을 작성하는 메서드**
 - 테두리의 기본 값은 검정색
 - fillStyle() 메서드를 사용해 변경할 수 있음

strokeRect(x좌표, y좌표, 너비, 높이)

strokeRect(100, 100, 50, 50)

- 메서드의 인자는 fillRect() 메서드와 동일함
- 너비(w)와 높이(h)가 모두 0인 경우
 - : 사각형이 그려지지 않음
- 너비(w) 또는 높이(h)가 0인 경우
 - : 시작 좌표가 (x, y) 일 때, (x, y) 좌표부터 (x+w, y+h)좌표까지의 직선이 그려짐





■ clearRect() 메서드

- **클리핑 영역을 지정하는 사각형을 작성하는 메서드**
 - 생성된 클리핑 영역과 중복되는 부분의 내용이 모두 지워짐

```
clearRect( x좌표, y좌표, 너비, 높이 )
```

```
clearRect( 100, 100, 50, 50 )
```

- 메서드의 인자는 fillRect() 메서드와 동일함
- 너비(w)나 높이(h) 중 하나가 0이거나 모두 다 0인 경우
 - : 어떠한 작업도 하지 않음



도형 요소의 작성 (사각형)

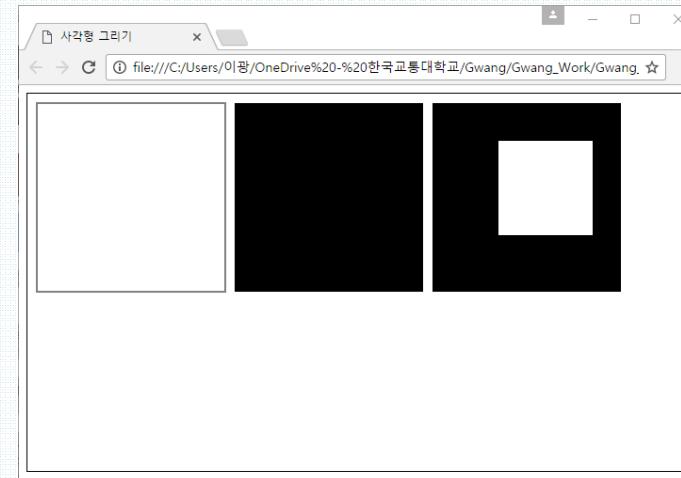


▣ 사각형 작성의 예

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <title> 사각형 그리기 </title>
    <script type="text/javascript">
        function rect()
        {
            var canvas = document.getElementById('canvas');
            context = canvas.getContext('2d');
            context.strokeRect(10,10,200,200);
            context.fillRect(220,10,200,200);
            context.fillRect(430,10,200,200);
            context.clearRect(500,50,100,100);
        }
    </script>
</head>
<body onload="rect();">

<canvas id="canvas" width="700" height="400" style="border:solid 1px #000000">
    canvas 사용하기
</canvas>

</body>
</html>
```





▣ 패스와 서브패스

- **패스(Path)**란 각 도형들을 이루는 선들의 집합을 의미
- **서브 패스(sub-path)**란 패스를 이루는 각각의 선을 의미
 - 하나의 패스는 여러 개의 서브 패스로 구성될 수 있음
- **선이나 도형 그리는 단계**

1. 패스를 초기화 함

: 새로운 패스를 시작함(beginPath() 메서드를 사용)

2. 패스를 지정함

: 생성된 패스 내에 서브패스들을 생성함

3. 패스에 효과를 지정함

: 지정한 패스를 그리기 메서드나 채우기 메서드를 사용하여 선이나 도형을 그림
: 생성된 서브 패스에 효과를 지정

4. 지정한 패스를 닫음

: 패스를 닫고 새로운 패스를 완성함 (closePath() 메서드를 사용)





▣ beginPath() 메서드

- 이전의 패스를 초기화하고 새로운 패스를 시작하는 메서드

```
context.beginPath()
```

- 패스를 생성하는 단계에서 최초로 수행
- 패스 당 하나의 beginPath() 메서드를 사용

▣ moveTo() 메서드

- 새로운 서브패스의 시작점을 생성하는 메서드

```
context.moveTo(x좌표, y좌표)
```

- x좌표, y좌표는 새로운 서브패스의 좌표를 의미
- 하나의 서브패스 내에서 하나의 moveTo() 메서드가 사용됨





▣ **lineTo()** 메서드

- **새로운 서브패스 좌표를 추가하며 이전에 추가된 좌표와 연결하는 메서드**

```
context.lineTo( x좌표, y좌표 )
```

- 지정된 x좌표, y좌표를 새로운 패스에 추가
 - : 직전에 추가되었던 좌표와 연결
- 연결만 할 뿐 라인을 그리지는 않음





■ stroke() 메서드

- 현재 생성되어 연결된 서브패스에 실제 라인을 그리는 메서드

```
context.stroke()
```

- stroke() 메서드가 수행되어야 패스 전체에 존재하는 서브패스에 실제 라인이 그려짐
 - : 모든 서브패스를 생성한 다음 실제 라인을 생성할 때 사용됨
- 그리기 색의 기본 값은 검정색이며, strokeStyle() 메서드를 사용해 색상 변경 가능

■ fill() 메서드

- 지정된 서브패스를 다각형으로 인식하고 내부를 색으로 채움

```
context.fill()
```

- 채워지는 기본 색은 검정색임
- fillStyle() 메서드를 사용해 채워지는 색상을 변경할 수 있음





■ closePath() 메서드

- 현재의 패스를 완성하고 닫는 기능을 수행하는 메서드

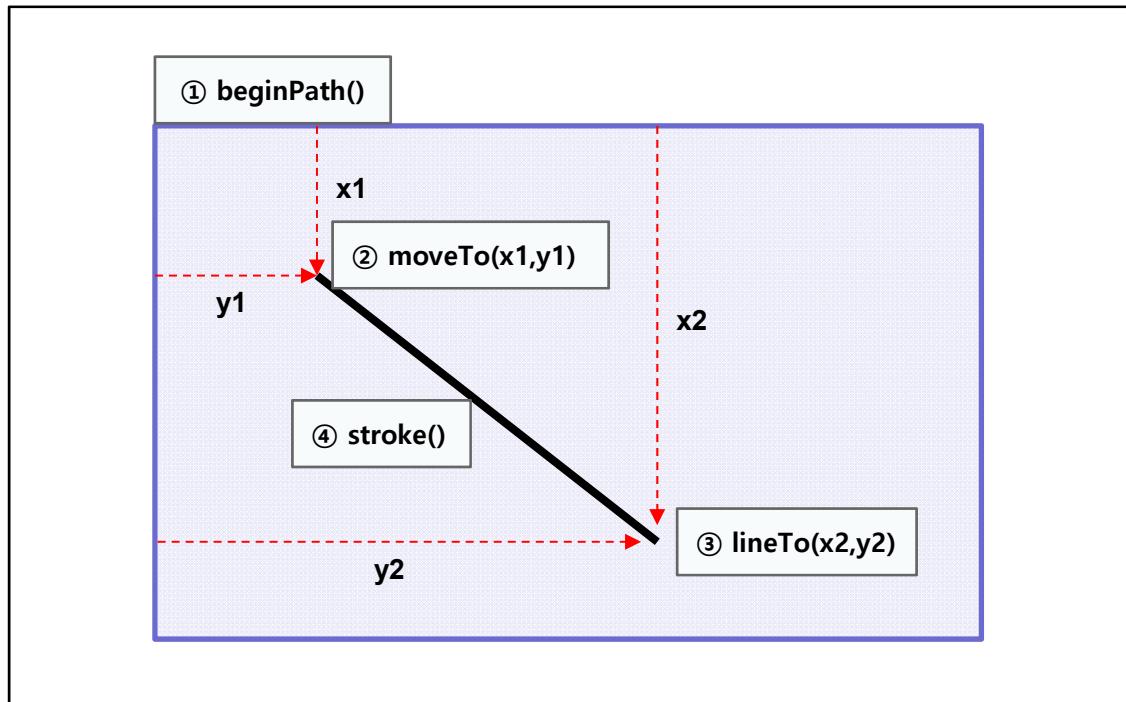
```
context.closePath()
```





■ 선(line) 작성

- `moveTo()`, `lineTo()`, `stroke()` 메서드를 사용해 선 그리기 수행 가능



도형 요소의 작성 (선과 다각형 작성)



■ 선(line) 작성의 예 - 1

```
<!DOCTYPE html>
<html>
<head>
    <script type="text/javascript">
        function line()
        {
            var canvas = document.getElementById('canvas');
            context = canvas.getContext('2d');

            context.beginPath();
            context.moveTo(10, 10);
            context.lineTo(100, 100);
            context.stroke();
        }
    </script>
</head>

<body onload="javascript:line()">
    <canvas id="canvas" width="300" height="300" style="border:solid 1px #000000">
        canvas 사용하기
    </canvas>
</body>
</html>
```



도형 요소의 작성 [선과 다각형 작성]



■ 선(line) 작성의 예 - 2

```
<!DOCTYPE html>
<html>
<head>
    <script type="text/javascript">
        function line()
        {
            var canvas = document.getElementById('canvas');
            context = canvas.getContext('2d');

            context.beginPath();
            context.moveTo(10, 10);
            context.lineTo(100, 50);
            context.lineTo(50, 100);
            context.stroke();
        }
    </script>
</head>

<body onload="javascript:line()">
    <canvas id="canvas" width="300" height="300" style="border:solid 1px #000000">
        canvas 사용하기
    </canvas>
</body>
</html>
```



도형 요소의 작성 (선과 다각형 작성)



■ 선(line) 작성의 예 - 3

```
<!DOCTYPE html>
<html>
<head>
    <script type="text/javascript">
        function line()
        {
            var canvas = document.getElementById('canvas');
            context = canvas.getContext('2d');

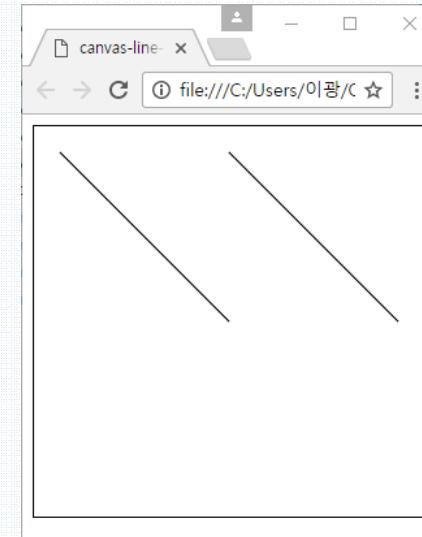
            context.beginPath();

            context.moveTo(20, 20);
            context.lineTo(150, 150);

            context.moveTo(150, 20);
            context.lineTo(280, 150)

            context.stroke();
        }
    </script>
</head>

<body onload="javascript:line()">
    <canvas id="canvas" width="300" height="300" style="border:solid 1px #000000">
        canvas 사용하기
    </canvas>
</body>
</html>
```





■ rect() 메서드

- 지정된 x, y좌표와 너비 그리고 높이를 사용해 사각 패스를 생성하는 메서드

```
context.rect( x좌표, y좌표, 너비, 높이 )
```

- 사각형을 작성하는 메서드들과 동일한 인자를 가짐
- 지정된 인자를 사용해 서브패스만을 형성할 뿐 사각형을 그리지는 않음
 - : 실제 사각형을 그리거나 채우기 위해서는 stroke() 메서드나 fill() 메서드를 사용해야 함



도형 요소의 작성 [선과 다각형 작성]



■ rect() 메서드의 예

```
<!DOCTYPE html>
<html>
<head>
    <title> 선 그리기 </title>
    <script type="text/javascript">
        function rect()
        {
            var canvas = document.getElementById('canvas');
            context = canvas.getContext('2d');

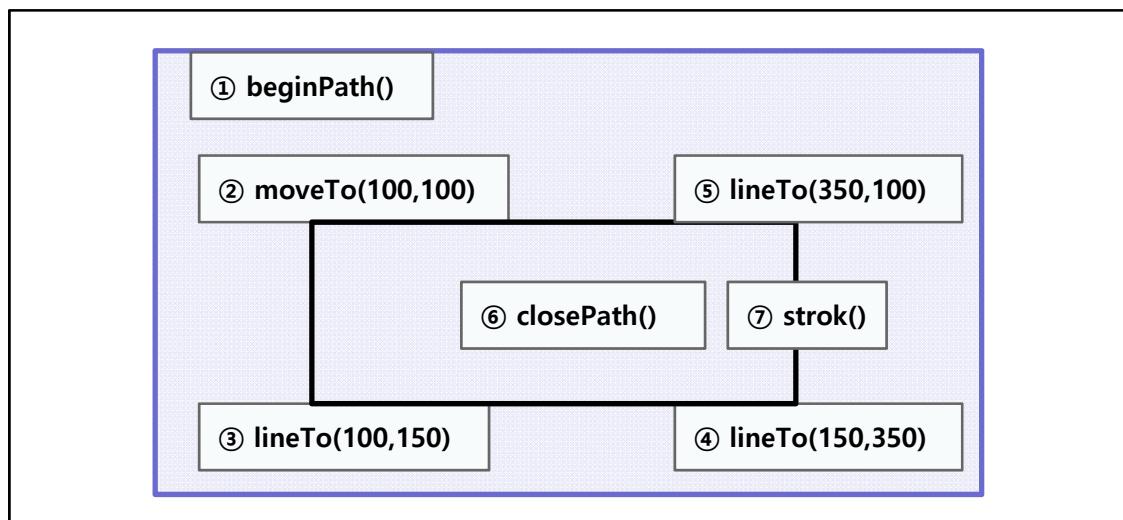
            context.beginPath();
            context.rect(50,50,300,150);
            context.stroke();
        }
    </script>
</head>
<body onload="rect();">
    <canvas id="canvas" width="400" height="300" style="border:solid 1px #000000"> canvas 사용하기</canvas>
</body>
</html>
```





▣ 다각형(polygon) 작성

- 선을 그리기 위한 `moveTo()`, `lineTo()`, `stroke()` 메서드에 `closePath()` 메서드를 추가해 다각형을 완성할 수 있음
 - `moveTo()`, `lineTo()` 메서드 : 서브패스를 생성
 - `closePath()` : 서브패스를 닫아 다각형을 완성
 - `stroke()`, `fill()` 메서드 : 라인을 그리거나 색을 채움
- `closePath()` 메서드를 사용해 서브패스를 닫은 다음 그리기를 수행해야 함

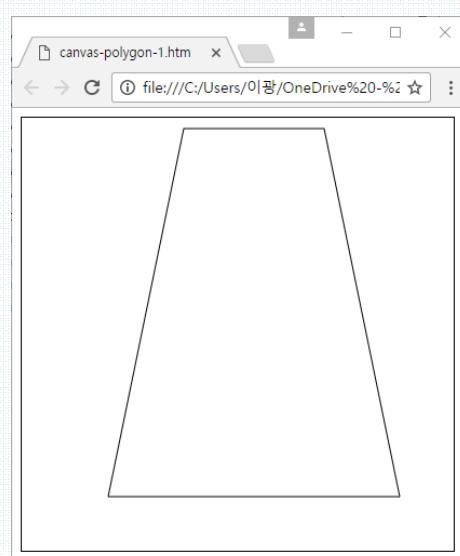


도형 요소의 작성 [선과 다각형 작성]



□ 다각형 작성의 예 - 1

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8" />
<script type="text/javascript">
    function polygon()
    {
        var canvas = document.getElementById('canvas');
        context = canvas.getContext('2d');
        context.beginPath();
        context.moveTo(150,10);
        context.lineTo(80,350);
        context.lineTo(350,350);
        context.lineTo(280,10);
        context.closePath();
        context.stroke();
    }
</script>
</head>
<body onload="polygon();">
<canvas id="canvas" width="400" height="400" style="border:solid 1px #000000">
    canvas 사용하기
</canvas>
</body>
</html>
```

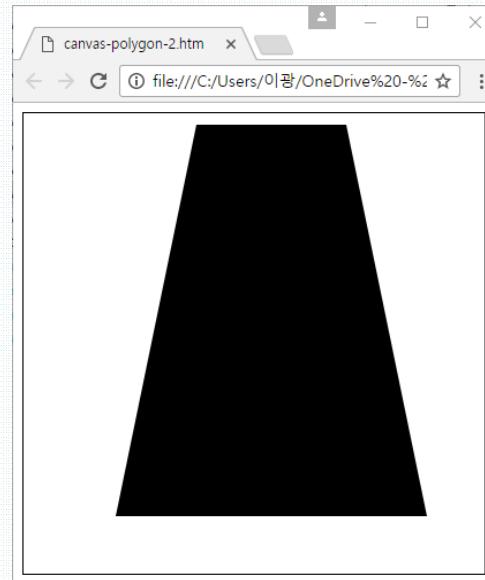


도형 요소의 작성 [선과 다각형 작성]



▣ 다각형 작성의 예 - 2

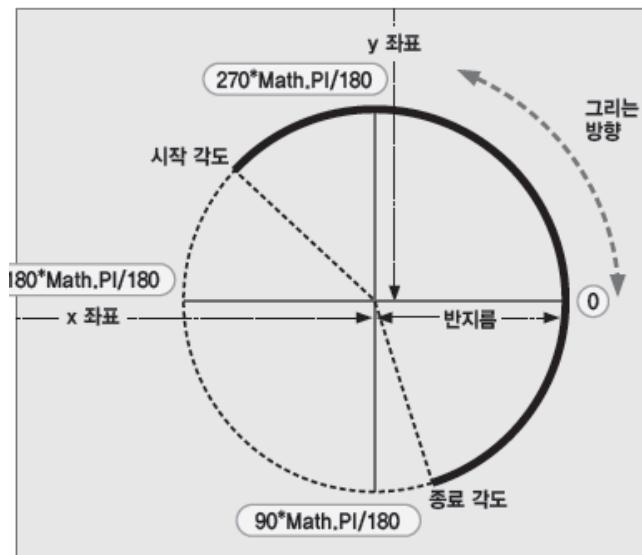
```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8" />
<script type="text/javascript">
    function polygon()
    {
        var canvas = document.getElementById('canvas');
        context = canvas.getContext('2d');
        context.beginPath();
        context.moveTo(150,10);
        context.lineTo(80,350);
        context.lineTo(350,350);
        context.lineTo(280,10);
        context.closePath();
        context.fill();
    }
</script>
</head>
<body onload="polygon();">
<canvas id="canvas" width="400" height="400" style="border:solid 1px #000000">
    canvas 사용하기
</canvas>
</body>
</html>
```





■ 원과 호 작성

- 원과 호를 그리기 위해서는 `arc()` 메서드를 사용
 - `arc()` 메서드에는 시작 x, y 좌표, 반지름, 시작 각도, 종료 각도, 그리는 방향을 인자 값으로 지정
 - : 방향이 true일 경우 : 반시계방향
 - : 방향이 false일 경우 : 시계향
- 시작 각도와 종료 각도의 단위는 도(degree)를 사용하지 않고 라디안(radian)을 사용.
 - 라디안 단위는 도에 `Math.PI/180`을 곱한 값





▣ arc() 메서드

- 원과 호를 작성하기 위한 메서드

```
arc( x좌표, y좌표, 반지름, 시작각도, 종료각도, 그리는방향 )
```

속성 값	설명
x좌표, y좌표	원이나 호의 중심 좌표를 의미
반지름	원이나 호의 반지름을 의미 (단위는 픽셀을 사용)
시작각도	원이나 호의 시작 각도를 의미 라디안으로 표현해야 함 (각도에 (Math.PI/180)을 곱해 표현)
종료각도	원이나 호의 종료 각도를 의미 라디안으로 표현해야 함 (각도에 (Math.PI/180)을 곱해 표현)
그리는 방향	원이라 호를 그리는 방향을 의미 true일 경우 : 반시계 방향, false일 경우 : 시계방향

- 원을 그릴 경우 시작각도와 종료각도는 같아야 함
- 원을 그릴 경우 그리는 방향은 true나 false이여도 무관함

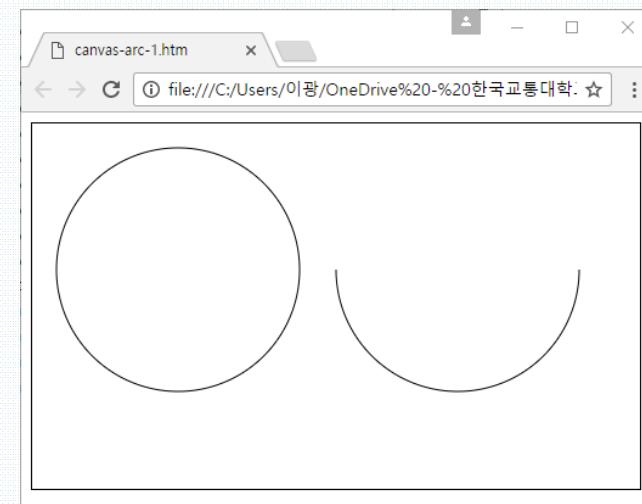




▣ arc() 메서드의 예 - 1

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <script type="text/javascript">
        function arc()
        {
            var canvas = document.getElementById('canvas');
            context = canvas.getContext('2d');
            context.beginPath();
            context.arc(120,120,100,0,360*Math.PI/180,false);
            context.stroke();

            context.beginPath();
            context.arc(350,120,100,0,180*Math.PI/180,false);
            context.stroke();
        }
    </script>
</head>
<body onload="arc();">
    <canvas id="canvas" width="500" height="300" style="border:solid 1px #000000">
        canvas 사용하기
    </canvas>
</body>
</html>
```





▣ arc() 메서드의 예 - 2

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8" />
<script type="text/javascript">
function arc()
{
    var canvas = document.getElementById('canvas');
    context = canvas.getContext('2d');

    context.beginPath();
    context.arc(120,120,100,90*Math.PI/180,360*Math.PI/180,false);
    context.stroke();

    context.beginPath();
    context.arc(120,120,100,90*Math.PI/180,360*Math.PI/180,true);
    context.fill();

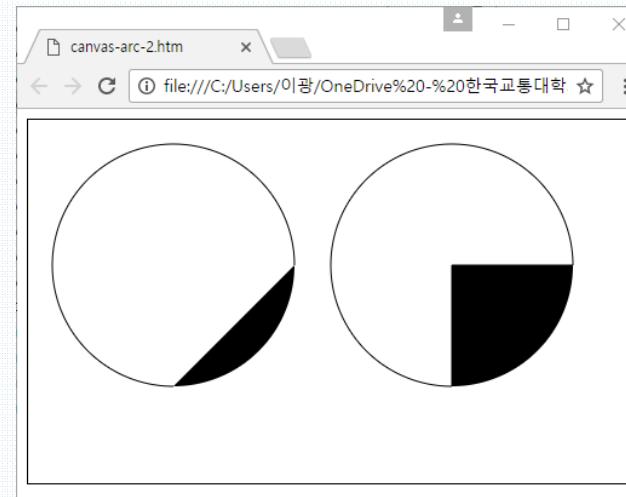
    context.beginPath();
    context.moveTo(350,120);
    context.arc(350,120,100,90*Math.PI/180,360*Math.PI/180,false);
    context.closePath();
    context.stroke();
```



도형 요소의 작성 [원과 호의 작성]



```
context.beginPath();
context.moveTo(350,120);
context.arc(350,120,100,90*Math.PI/180,360*Math.PI/180,true);
context.closePath();
context.fill();
}
</script>
</head>
<body onload="arc();">
<canvas id="canvas" width="500" height="300" style="border:solid 1px #000000">
    canvas 사용하기
</canvas>
</body>
</html>
```

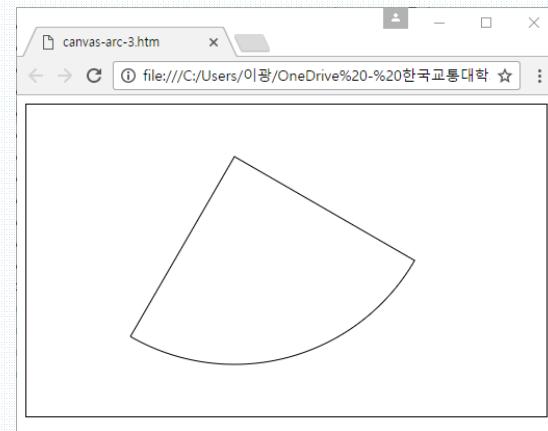


도형 요소의 작성 [원과 호의 작성]



▣ arc() 메서드의 예 - 3

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <script type="text/javascript">
        function arc()
        {
            var canvas = document.getElementById('canvas');
            context = canvas.getContext('2d');
            context.beginPath();
            context.moveTo(200,50);
            context.arc(200,50,200,30*Math.PI/180,120*Math.PI/180,false);
            context.closePath()
            context.stroke();
        }
    </script>
</head>
<body onload="arc();">
    <canvas id="canvas" width="500" height="300" style="border:solid 1px #000000">
        canvas 사용하기
    </canvas>
</body>
</html>
```



도형 요소의 작성 [직선과 인접한 호 작성]



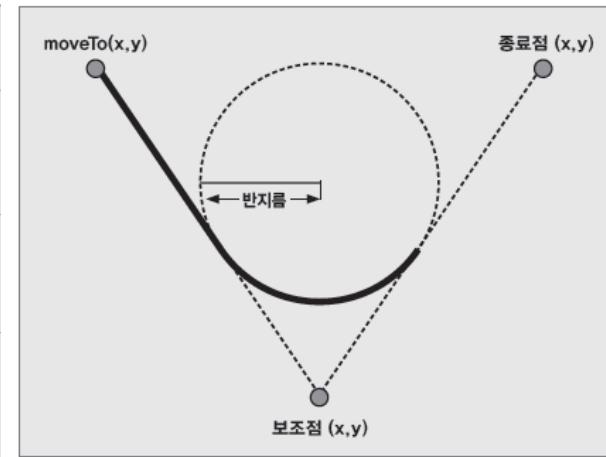
▣ arcTo() 메서드

- 직선에 인접한 호를 작성하기 위한 메서드

arcTo(보조점x, 보조점y, 종료점x, 종료점y, 반지름)

- 보조점과 종료점을 지정해 선을 완성하고 반지름을 이용해 호의 모양을 결정
- 선의 시작점은 moveTo() 메서드를 사용

속성 값	설명
보조점x, 보조점y	moveTo() 메서드로 지정한 좌표부터 서브패스를 형성하기 위한 좌표
종료점x, 종료점y	보조점의 좌표와 서브패스를 형상하기 위한 좌표
반지름	보조점과 종료점으로 생성된 두 선에 인접하는 원의 반지름



도형 요소의 작성 [직선과 인접한 호 작성]



▣ arcTo() 메서드의 예 - 1

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <script type="text/javascript">
        function arcTo()
        {
            var canvas = document.getElementById('canvas');
            context = canvas.getContext('2d');
            context.beginPath();
            context.moveTo(100,100);
            context.arcTo(400,100,400,300,100);
            context.stroke();
        }
    </script>
</head>
<body onload="arcTo();">
    <canvas id="canvas" width="500" height="300" style="border:solid 1px #000000">
        canvas 사용하기
    </canvas>
</body>
</html>
```

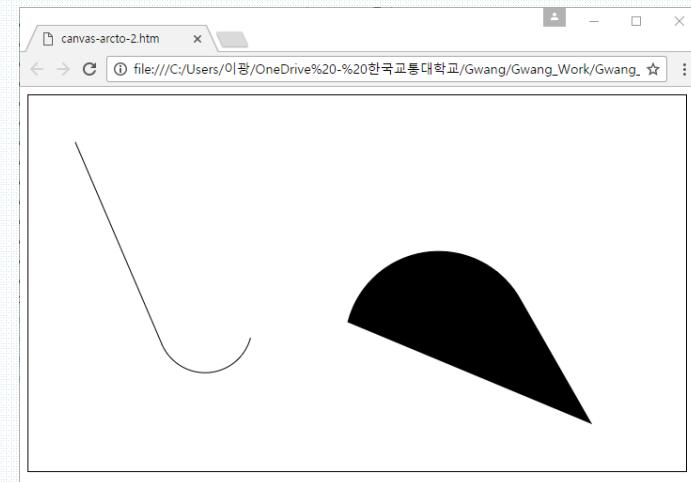


도형 요소의 작성 [직선과 인접한 호 작성]



▣ arcTo() 메서드의 예 - 2

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <script type="text/javascript">
        function arcTo()
        {
            var canvas = document.getElementById('canvas');
            context = canvas.getContext('2d');
            context.beginPath();
            context.moveTo(50,50);
            context.arcTo(200,400,300,10,50);
            context.stroke();
            context.beginPath();
            context.moveTo(600,350);
            context.arcTo(400,0,300,400,100);
            context.fill();
        }
    </script>
</head>
<body onload="arcTo();">
    <canvas id="canvas" width="700" height="400" style="border:solid 1px #000000">
        canvas 사용하기
    </canvas>
</body>
</html>
```



도형 요소의 작성 [베지에 곡선 작성]



▣ quadraticCurveTo() 메서드

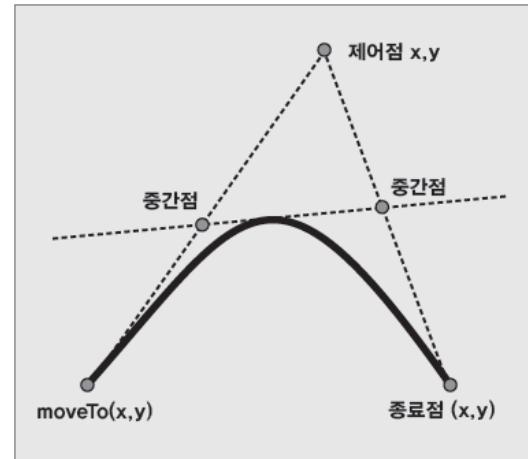
- 2차 베지에 곡선을 생성하기 위한 메서드

- 베지어 곡선(Bezier Curve)

- : n개의 점을 기준으로 만들 수 있는 (n-1)차 곡선을 의미
 - : 2차 베지에 곡선은 시작점과 종료점이 존재하고 하나의 제어점이 존재

quadraticCurveTo(제어점x, 제어점y, 종료점x, 종료점y)

속성 값	설명
보조점x, 보조점y	moveTo() 메서드로 지정한 좌표부터 서브 패스를 형성하기 위한 좌표
종료점x, 종료점y	보조점의 좌표와 서브패스를 형상하기 위한 좌표

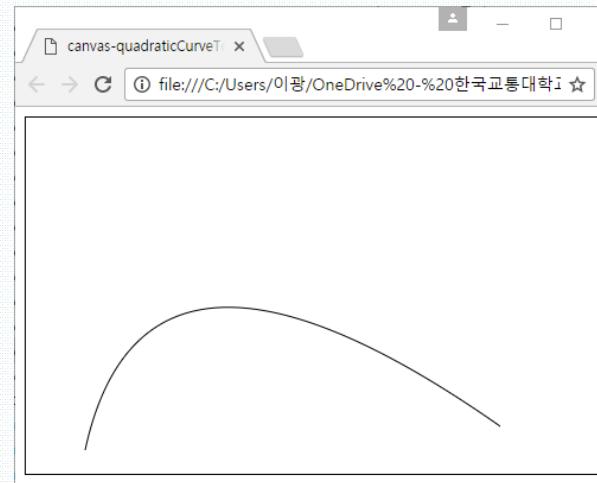


도형 요소의 작성 (베지에 곡선 작성)



▣ quadraticCurveTo() 메서드의 예

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <script type="text/javascript">
        function quad()
        {
            var canvas = document.getElementById('canvas');
            context = canvas.getContext('2d');
            context.beginPath();
            context.moveTo(50,280);
            context.quadraticCurveTo(100,50,400,260);
            context.stroke();
        }
    </script>
</head>
<body onload="quad();">
    <canvas id="canvas" width="500" height="300" style="border:solid 1px #000000">
        canvas 사용하기
    </canvas>
</body>
</html>
```



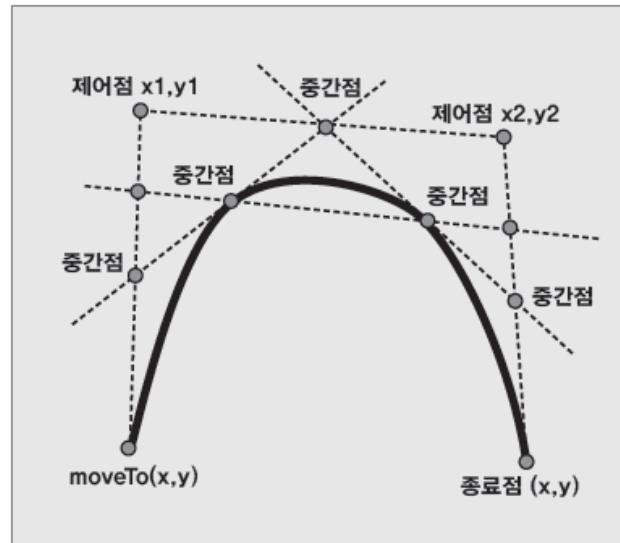


■ bezierCurveTo() 메서드

- 3차 베지에 곡선을 생성하기 위한 메서드

```
bezierCurveTo( 제어점x1, 제어점y1, 제어점x2, 제어점y2, 종료점x, 종료점y )
```

- 3차 베지에 곡선은 2개의 제어점을 가짐
 - : 총 5개의 보조선이 생성되어 이를 기준으로 곡선이 생성



도형 요소의 작성 (베지에 곡선 작성)



■ bezierCurveTo() 메서드의 예 - 1

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />

    <script type="text/javascript">
function bez()
{
    var canvas = document.getElementById('canvas');
    context = canvas.getContext('2d');
    context.beginPath();
    context.moveTo(100,220);
    context.bezierCurveTo(60,50,320,50,400,280)
    context.stroke();
}
</script>

</head>
<body onload="bez();">
<canvas id="canvas" width="500" height="300" style="border:solid 1px #000000">
    canvas 사용하기
</canvas>
</body>
</html>
```



도형 요소의 작성 (베지에 곡선 작성)



■ bezierCurveTo() 메서드의 예 - 2

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <script type="text/javascript">
        function bez()
        {
            var canvas = document.getElementById('canvas');
            context = canvas.getContext('2d');
            context.beginPath();
            context.moveTo(50,200);
            context.bezierCurveTo(100,50,350,200,400,50)
            context.stroke();
        }
    </script>
</head>
<body onload="bez();">
    <canvas id="canvas" width="500" height="300" style="border:solid 1px #000000">
        canvas 사용하기
    </canvas>
</body>
</html>
```





도형 스타일 지정

점선(dashed line) 작성



▣ **setLineDash()** 메서드

- **점선을 그리기 위해 사용되는 메서드**
 - 점선을 이루는 일정한 패턴을 생성할 수 있음
 - 일정한 간격으로 점선을 이루는 내부선과 각 내부선 사이의 간격을 지정할 수 있음

setLineDash([n])

: 점선을 이루는 내부선과 각각의 내부선 사이의 간격을 모두 n으로(픽셀) 지정

setLineDash([n1, n2])

: 점선을 이루는 내부선의 길이를 n1으로, 내부선 사이의 간격을 모두 n2로(픽셀) 지정

setLineDash([n1, n2, n3, n4])

: 첫 번째 내부선의 길이를 n1으로, 그 다음 내부선과의 간격을 n2로 지정
: 두 번째 내부선의 길이를 n2으로, 그 다음 내부선과의 간격을 n3로 지정
: 위의 과정이 반복되는 점선 패턴을 생성



점선(dashed line) 작성



▣ setLineDash() 메서드의 예

```
<!DOCTYPE html>
<html>
<head>
    <title> 선 그리기 </title>
    <script type="text/javascript">
        function linedash()
        {
            var canvas = document.getElementById('canvas');
            context = canvas.getContext('2d');

            context.beginPath();
            context.setLineDash([10]);
            context.rect(30,15,100,100);
            context.stroke();
            context.closePath();

            context.beginPath();
            context.setLineDash([10,5]);
            context.moveTo(145,115);
            context.lineTo(195,15);
            context.lineTo(245,115);
            context.closePath();
            context.stroke();
            context.closePath();
        }
    </script>
</head>
<body>
    <div>
        <canvas id="canvas" width="200" height="100"></canvas>
    </div>
</body>
</html>
```



점선(dashed line) 작성

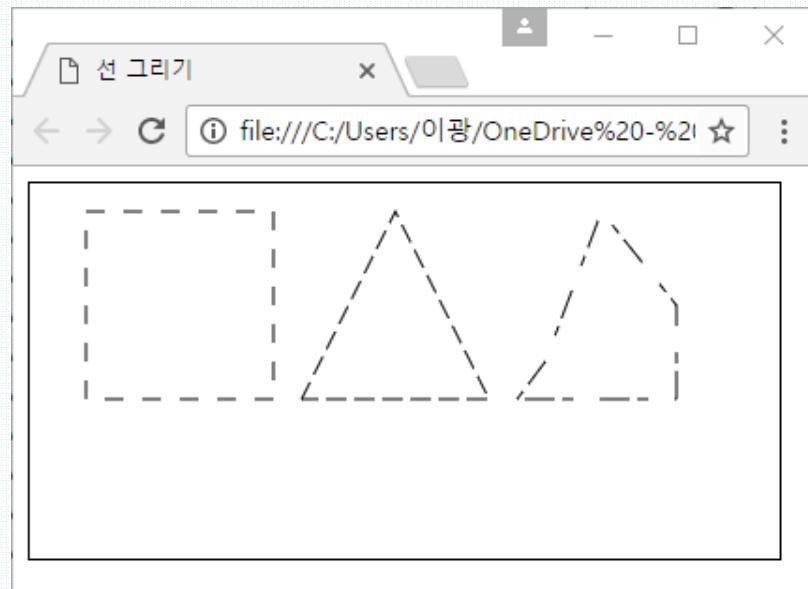


```
context.beginPath();
context.fillStyle = 'lightblue';
context.setLineDash([5,5,15,15]);
context.moveTo(260,115);
context.lineTo(275,95);
context.lineTo(305,15);
context.lineTo(345,65);
context.lineTo(345,115);
context.closePath();
context.stroke();
context.closePath();

}

</script>
</head>

<body onload="linedash();">
<canvas id="canvas" width="400" height="200" style="border:solid 1px #000000">
    canvas 사용하기
</canvas>
</body>
</html>
```





■ **lineWidth 속성**

- 패스의 두께를 지정하기 위한 속성

lineWidth = 선의두께

- 선의 두께 지정을 위해 픽셀을 사용

■ **strokeStyle 속성**

- 패스의 색상을 지정하기 위한 속성

strokeStyle = “선의색상”

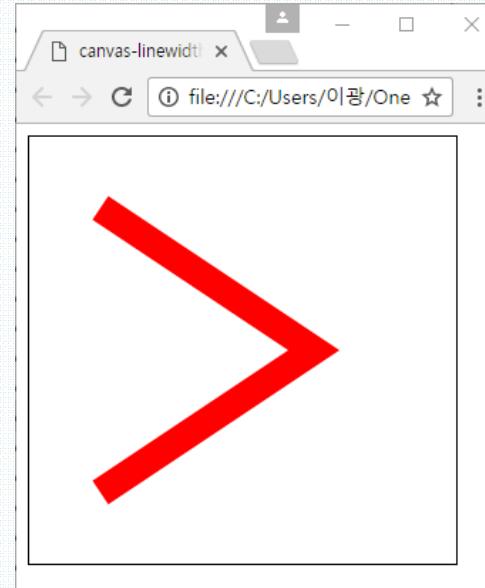
- 선의 색상 지정을 위해 색상이름, 16진수 표현, rgb(), rgba(), hsl(), hsla() 사용 가능





■ lineWidth 속성, strokeStyle 속성의 예

```
<!DOCTYPE html>
<html>
<head>
<script type="text/javascript">
function line()
{
    var canvas = document.getElementById('canvas');
    context = canvas.getContext('2d');
    context.beginPath();
    context.moveTo(50,50);
    context.lineTo(200,150);
    context.lineTo(50,250);
    context.lineWidth=20;
    context.strokeStyle="red";
    context.stroke();
}
</script>
</head>
<body onload="line();">
<canvas id="canvas" width="300" height="300" style="border:solid 1px #000000">
    canvas 사용하기
</canvas>
</body>
</html>
```





■ lineCap 속성

- 선의 끝 모양을 지정하기 위한 속성

lineCap = “끝모양”

끝 모양 값	설명
butt	선의 끝 모양을 변경하지 않고 그대로 표현(기본값)
round	선 두께에 해당하는 지름을 갖는 반원을 끝에 추가
square	선 두께에 해당하는 너비를 갖는 사각형의 반을 끝에 추가



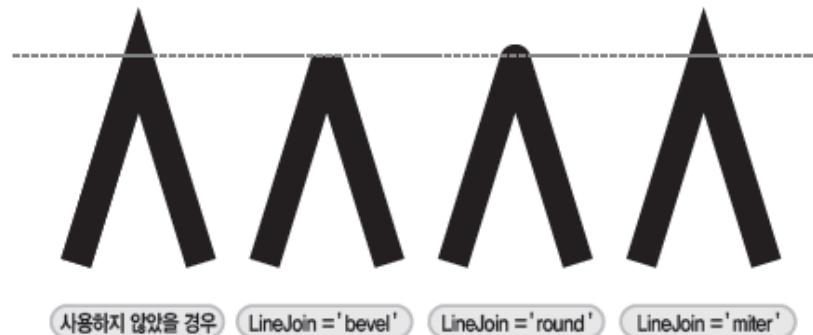


■ lineJoin 속성

- 선들이 연결된 모양을 지정하기 위한 속성

lineJoin = “연결모양”

연결 모양 값	설명
bevel	선이 연결된 부분을 깎아서 사선으로 표현
miter	선이 연결된 부분을 날카롭게 표현 (기본값)
round	선이 연결된 부분을 둥근 원으로 표현

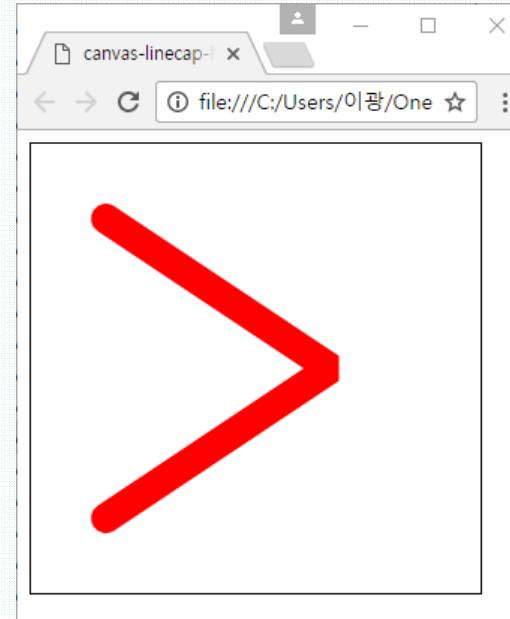


선의 끝부분과 연결부분 지정



■ lineCap 속성, lineJoin 속성의 예

```
<!DOCTYPE html>
<html>
<head>
    <script type="text/javascript">
        function line()
        {
            var canvas = document.getElementById('canvas');
            context = canvas.getContext('2d');
            context.beginPath();
            context.moveTo(50,50);
            context.lineTo(200,150);
            context.lineTo(50,250);
            context.lineWidth=20;
            context.strokeStyle="red";
            context.lineCap="round";
            context.lineJoin="bevel"
            context.stroke();
        }
    </script>
</head>
<body onload="line();">
    <canvas id="canvas" width="300" height="300" style="border:solid 1px #000000">
        canvas 사용하기
    </canvas>
</body>
</html>
```





■ fillStyle 속성

- 도형을 채우기 위한 색상을 지정하기 위한 속성

`fillStyle = “채우기색”`

- 채우기 색은 색상이름, 16진수 표현, `rgb()`, `rgba()`, `hsl()`, `hsla()` 사용 가능

■ globalAlpha 속성

- 도형을 채우는 색상의 투명도를 지정하기 위한 속성

`globalAlpha = 투명도`

- 투명도는 0부터 1까지 지정가능 (0에 가까울수록 투명함)





■ fillStyle 속성, globalAlpha 속성의 예

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <script type="text/javascript">
        function rect()
        {
            var canvas = document.getElementById('canvas');
            context = canvas.getContext('2d');

            context.lineWidth = 30;
            context.strokeStyle = 'blue'
            context.strokeRect(50, 50, 200, 200);
            context.fillStyle = 'red';
            context.globalAlpha = '0.5';
            context.fillRect(50, 50, 200, 200);
        }
    </script>
</head>

<body onload="rect();">
    <canvas id="canvas" width="400" height="300" style="border:solid 1px #000000">
        canvas 사용하기
    </canvas>
</body>
</html>
```

