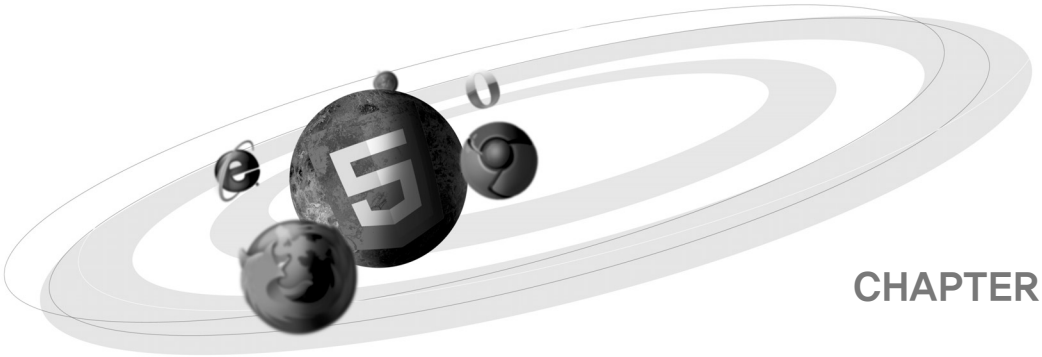

CHAPTER 2

Canvas 요소로 그림 그리기

HTML5의 Canvas 요소는 많은 사람들이 주목하고 있는 기능 중 하나로서 그래픽을 화면에 표시할 때 사용된다. HTML5에서 Canvas 요소의 생성은 아래처럼 아주 간단하다.

```
<canvas height="yy" width="xxx">
</canvas>
```

이 코드가 Canvas 요소를 생성하는 데 필요한 전부이다. 그렇다면 이 요소 안에 그림을 그려 넣기 위해서는 어떻게 해야 할까? 별로 고민할 필요 없이 자바스크립트를 사용하면 된다. 이 장에서는 바로 자바스크립트로 그림을 그리는 방법에 대해서 살펴볼 것이다.

Canvas 요소는 선, 둥근 모양, 복잡한 형태의 도형, 이미지, 텍스트 외에도 많은 것들을 그릴 수 있도록 풍부한 기능을 제공한다. 이제부터 Canvas 요소에 대해서 좀 더 자세히 살펴보자.

Canvas 요소 시작하기

기술적인 측면에서 말하자면, HTML5의 Canvas 요소는 다음의 명세와 같이 아주 간단하다.

- 요소: 〈Canvas〉
- 시작 태그 필요: Yes
- 끝 태그 필요: Yes
- 필요한 애트리뷰트: Height, Width
- 지원 브라우저: 크롬, 파이어폭스, 오페라, 사파리

이 장에서 설명할 Canvas 요소에 대한 내용은 실제로 자바스크립트와 함께 진행될 것이며, 그 결과로 그림 2.1에 나타난 도형들을 예제로 그려볼 것이다.

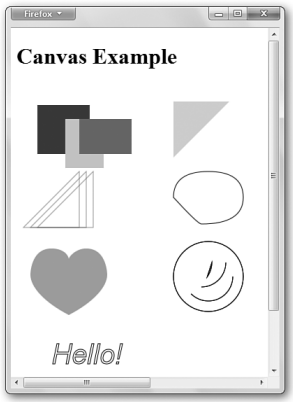

그림 2.1 파이어폭스로 실행한 Canvas 요소 예제

Canvas 요소를 동작시키기 위해선 자바스크립트를 사용해야 한다. 그렇기 때문에 예제 코드를 시작하기에 앞서 우리에게 필요한 내용들을 간단히 살펴보기로 하자.

Canvas API 살펴보기

W3C는 Canvas 요소를 위해 필요한 애플리케이션 프로그래밍 인터페이스(API)를 마련하고, 내장 함수의 이름과 이 함수를 어떻게 사용하는지에 대해 명시하고 있다.

http://dev.w3.org/html5/canvas-api/canvas-2d-api.html을 통해 전체 Canvas의 API를 확인해 볼 수 있는데, 그 중에서 우리는 가장 중요한 함수 몇 가지를 이 장을 통해서 살펴볼 것이다. W3C API 명세 중에는 Canvas 요소의 두 가지 애트리뷰트와 지원되는 자바스크립트 함수가

나열돼 있다(여기에서 두 애트리뷰트는 HTML의 애트리뷰트가 아니고 자바스크립트에서 사용되는 canvas1.fillStyle = xxxx와 같은 요소의 애트리뷰트를 말한다). 그렇기 때문에 첫 번째 애트리뷰트에 원하는 값을 할당한 후 Canvas 요소가 어떻게 실행되는지를 살펴보면 된다. 예제 코드에서는 첫 번째로 fillStyle 애트리뷰트를 사용한 후에 fillRect 함수를 사용해서 사각형을 그릴 것이다.

```
canvas1.fillStyle =xxxx
canvas1.fillRect(xx, xx, xx, xx);
```

API의 모든 항목은 자신의 데이터 타입을 가리키는 어휘를 갖는데, 그 중 부동소수점을 가리키는 float를 가장 흔한 예로 들 수 있다. 다음은 W3C에 명시된 대표적인 타입들을 나열한 것이다.

- any: 이 애트리뷰트는 모든 타입이 될 수 있다.
- DOM(Document Object Model) 문자열을 의미하는 DOMString: 따옴표로 감싸인 문자열
- float: 부동소수점을 나타낸다.

그럼 이제 Canvas 요소에는 어떤 애트리뷰트와 함수들이 존재하는지 살펴보자.

스타일

Canvas에서는 그리는 도형을 채울 것인지 아닌지를 설정하기 위해 두 개의 애트리뷰트를 제공한다.

- 애트리뷰트 any fillStyle; // (디폴트 검은색)
- 애트리뷰트 any strokeStyle; // (디폴트 검은색)

선 스타일 설정

Canvas 요소에 선 스타일을 설정하기 위해서는 다음과 같은 자바스크립트 애트리뷰트를 사용하면 된다.

- 애트리뷰트 DOMString lineCap; // "butt", "round", "square" (디폴트 "butt")
- 애트리뷰트 DOMString lineJoin; // "miter", "round", "bevel" (디폴트 "miter")

- 애트리뷰트 float lineWidth; // (디폴트 1)
- 애트리뷰트 float miterLimit; // (디폴트 10)

그림자 적용하기

다음과 같이 애트리뷰트를 적용하면 Canvas 요소에 그림자를 추가할 수 있다.

- 애트리뷰트 float shadowBlur; // (디폴트 0)
- 애트리뷰트 DOMString shadowColor; // (디폴트 투명한 검은색)
- 애트리뷰트 float shadowOffsetX; // (디폴트 0)
- 애트리뷰트 float shadowOffsetY; // (디폴트 0)

사각형 그리기

다음은 사각형을 그릴 때 사용하는 함수들이다.

- clearRect(float x, float y, float w, float h);
- fillRect(float x, float y, float w, float h);
- strokeRect(float x, float y, float w, float h);

복잡한 도형 그리기

다음 함수들을 사용하면 Canvas 요소에 둥근 모양과 곡선을 비롯해 수많은 도형들을 그릴 수 있다.

- arc(float x, float y, float radius, float startAngle, float endAngle, boolean anticlockwise);
- arcTo(float x1, float y1, float x2, float y2, float radius);
- beginPath();
- bezierCurveTo(float cp1x, float cp1y, float cp2x, float cp2y, float x, float y);
- clip();
- closePath();
- fill();
- lineTo(float x, float y);
- moveTo(float x, float y);
- quadraticCurveTo(float cpx, float cpy, float x, float y);

- rect(float x, float y, float w, float h);
- stroke();
- booleanisPointInPath(float x, float y);

문자열 그리기

다음 애트리뷰트와 함수를 사용해서 Canvas에 문자를 쓸 수 있다.

- 애트리뷰트 DOMString font; // (디폴트 10px sans-serif체)
- 애트리뷰트 DOMString textAlign; // "start", "end", "left", "right", "center" (디폴트: "start")
- 애트리뷰트 DOMStringtextBaseline; // "top", "hanging", "middle", "alphabetic", "ideographic", "bottom" (디폴트: "alphabetic")
- fillText(DOMString text, float x, float y, optional float maxWidth);
- TextMetricsmeasureText(DOMString text);
- strokeText(DOMString text, float x, float y, optional float maxWidth);

이미지 그리기

다음 함수들을 사용해 이미지를 그릴 수 있다.

- drawImage(HTMLImageElement image, float dx, float dy, optional float dw, float dh);
- drawImage(HTMLImageElement image, float sx, float sy, float sw, float sh, float dx, float dy, float dw, float dh);
- drawImage(HTMLCanvasElement image, float dx, float dy, optional float dw, float dh);
- drawImage(HTMLCanvasElement image, float sx, float sy, float sw, float sh, float dx, float dy, float dw, float dh);
- drawImage(HTMLVideoElement image, float dx, float dy, optional float dw, float dh);
- drawImage(HTMLVideoElement image, float sx, float sy, float sw, float sh, float dx, float dy, float dw, float dh);

도형 변환

다음 함수들을 통해서 도형 회전이나 사이즈(크기) 변경, 도형 이동 등을 처리할 수 있다.

- rotate(float angle);

- scale(float x, float y);
- translate(float x, float y);

여기까지가 Canvas API의 개요라 할 수 있다. 그럼 이제 본격적으로 예제 코드를 작성하면서 Canvas 요소에 대해 살펴보자.

Canvas 예제 시작

Canvas 요소가 어떻게 동작하는지를 설명하기 위해 canvas.html이라는 파일을 하나 생성한다. 이 장 전체에서 설명한 내용들은 전부 canvas.html에 저장되며, 작성이 끝나고 난 후 파이어폭스를 사용해 파일을 실행하면 앞서 살펴본 그림 2.1과 같은 화면이 나타나는 것을 확인할 수 있다.

그럼 다음에서 설명하는 단계에 따라 canvas.html이라는 예제 파일을 작성해 보자.

1. 메모장과 같은 텍스트 편집기를 사용해 casnvas.html 파일을 생성한다.
2. <canvas> 요소를 생성하기 위해 아래와 같은 코드를 작성하고 자바스크립트 영역도 준비한다. 주의할 것은 loader라는 이름의 자바스크립트 함수를 생성해서 그 안에 테스트할 코드를 작성할 것인데, 이 함수는 브라우저가 <canvas> 요소를 완전히 로드한 후에 실행된다(세로로 연속해서 표시된 점은 어떤 코드가 위치할 것이라는 뜻을 보여주기 위해 사용한 것이니 예제 코드에는 추가하지 않는다).

```
<!DOCTYPE html>
<html>
  <head>
    <title>
      Canvas Example
    </title>

    <script type="text/javascript">
      function loader()
      {
        .
        .
```

```
        .
      }
    </script>
  </head>

  <body onload="loader()">
    <h1>Canvas Example</h1>
    <canvas id="canvas" width="600" height="500">
    </canvas>

  </body>
</html>
```

3. 다음과 같이 Canvas 객체를 생성하는 자바스크립트를 추가한다. 여기에서 생성한 Canvas 요소의 인스턴스는 자바스크립트 내에서 Canvas 요소에 접근할 때 사용될 것이다.

```
<!DOCTYPE html>
<html>
  <head>
    <title>
      Canvas Example
    </title>

    <script type="text/javascript">
      function loader()
      {
      var canvas = document.getElementById("canvas");
      var canvas1 = canvas.getContext("2d");
        .
        .
        .
      }
    </script>
  </head>

  <body onload="loader()">
    <h1>Canvas Example</h1>
    <canvas id="canvas" width="600" height="500">
    </canvas>

  </body>
</html>
```

4. canvas.html 파일을 저장한다.(**역자 주**_원서에서는 예제 코드를 윈도우의 WordPad를 사용해서 작성하기 때문에 파일 저장 시 RTF 포맷이 아닌 텍스트 포맷으로 저장하는 부분에 주의하라고 명시하고 있지만, 우리나라 개발자의 정서상 본 역서에서는 메모장으로 번역하기 때문에 파일 저장 시의 포맷에 대해서는 언급하지 않는다).

지금까지 예제 코드 작성을 위한 준비단계를 마쳤으니 이제 몇 개의 사각형을 그려보자.

사각형 그리기

strokeRect 함수를 사용하면 속이 빈 사각형을 그릴 수 있다.

- strokeRect(float x, float y, float w, float h);

그렇지 않고 속이 채워진 사각형을 그리고 싶으면, fillRect 함수를 사용하면 된다.

- fillRect(float x, float y, float w, float h);

이 함수를 사용할 때는 사각형의 왼쪽 상단 모서리의 좌표를 함수의 (x, y) 매개변수로 전달하고, 원하는 만큼의 넓이와 높이를 나머지 (w, h) 매개변수에 전달하면 된다. 한 가지 Canvas 요소에서 주의해야 할 점은, Canvas의 왼쪽 상단 모서리의 좌표 (0, 0)을 기준으로 x축의 값이 양수이면 x축의 위치는 왼쪽으로 움직이고, y축 역시 값이 양수일 경우에는 양수의 값만큼 y축의 위치는 아래로 움직이며, 모든 측정 단위는 픽셀(pixel)이다.

이번 예제 코드에서는 fillRect 함수를 사용할 것인데, fillRect 함수를 사용해서 속이 채워진 사각형을 그릴 경우에는 fillStyle 애트리뷰트를 사용해 사각형 안의 채울 색을 지정할 수 있다. 사각형 안을 채울 색은 rgba( ) 함수를 통해 지정한다. rgba( ) 함수는 빨강, 초록, 그리고 파랑 값을 나타내는 세 개의 매개변수와 가시성 여부를 가리키는 하나의 매개변수를 포함해 총 4개의 매개변수를 가진다(가시성을 구분 짓는 맨 마지막 매개변수의 값은 0과 1로 나뉘며, 이 값이 0일 경우는 사각형은 보이지 않고, 1일 경우에만 보이게 된다). rgba( ) 함수를 사용해 원하는 색을 지정하고 싶다면 빨강, 초록, 파랑을 가리키는 각각의 매개변수에 0부터 255 사이의 값을 함수에 전달하면 된다.

예를 들어, 파란색으로 속이 채워진 사각형을 canvas1에 그리고 싶을 경우에는 다음과 같이 코드를 작성하면 된다.

- canvas1.fillStyle = "rgba(0, 0, 200, 1)";

그럼 이번에는 다음과 같은 과정을 통해 서로 다른 색으로 속이 채워진 여러 개의 사각형을 그려보자.

1. 메모장과 같은 텍스트 편집기를 사용해서 canvas.html 파일을 연다.
2. 서로 다른 색을 가진 세 개의 사각형을 그리기 위해 아래의 코드를 추가한다.

```
<script type="text/javascript">
  function loader()
  {
  var canvas = document.getElementById("canvas");
  var canvas1 = canvas.getContext("2d");

  // Rectangles
  canvas1.fillStyle = "rgba(0, 0, 200, 1)";
  canvas1.fillRect(30, 30, 75, 70);

  canvas1.fillStyle = "rgba(200, 200, 0, 1)";
  canvas1.fillRect(70, 50, 55, 70);

  canvas1.fillStyle = "rgba(200, 0, 0, 1)";
  canvas1.fillRect(90, 50, 75, 50);
    .
    .
    .
  }
```

3. canvas.html 파일을 저장한다.

저장한 파일을 역시 파이어폭스를 통해 실행해 보면 그림 2.2와 같이 모두 세 개의 겹쳐진 사각형이 나타나는 것을 확인할 수 있다.

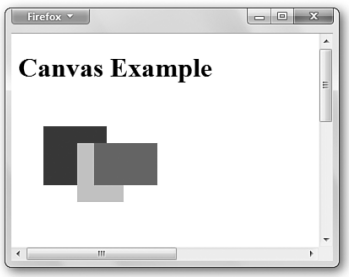

그림 2.2 여러 개의 사각형 그리기

선 도형 그리기

Canvas 컨트롤을 사용하면 선으로 이루어진 도형도 그릴 수 있다. 선으로 도형을 그리기 위해서는 맨 먼저 Canvas 컨트롤에게 지금 도형을 그리고 있다는 것을 알려주기 위해 beginPath() 함수를 호출한다. 그런 다음, 그리는 위치와 실제 선의 위치를 알려주기 위해 moveTo() 함수와 lineTo() 함수를 혼용해 사용한다.

이렇게 해서 선 도형 그리기를 마치고 나면, closePath() 함수를 호출해 지금까지 그린 선의 경로를 완성시키고 stroke() 함수를 호출해 결과를 화면에 그린다.

그럼 이제 지금까지 설명한 방법을 사용해서 세 개의 삼각형을 그려보자. 추가로, 여기에 앞서 설명한 방법을 적용해 Canvas의 storkeStyle 애트리뷰트에 빨간 색 값을 할당해 붉은 색으로 된 세 개의 삼각형을 그려보자.

- canvas1.strokeStyle = "rgba(200, 0, 0, 0.5)";

다음은 같은 과정을 거쳐 예제 코드를 완성한다.

1. 메모장과 같은 텍스트 편집기를 사용해 canvas.html 파일을 연다.
2. 세 개의 삼각형을 그리기 위해 아래와 같이 코드를 추가한다.

```
<script type="text/javascript">
  function loader()
  {
  var canvas = document.getElementById("canvas");
  var canvas1 = canvas.getContext("2d");
    .
    .
    .
  // Stroked triangles
  canvas1.beginPath();
  canvas1.strokeStyle = "rgba(200, 0, 0, 0.5)";
  canvas1.moveTo(110, 205);
  canvas1.lineTo(110, 125);
  canvas1.lineTo(30, 205);
  canvas1.closePath();
  canvas1.stroke();
```

```
canvas1.beginPath();
canvas1.moveTo(100, 205);
canvas1.lineTo(100, 125);
canvas1.lineTo(20, 205);
canvas1.closePath();
canvas1.stroke();

canvas1.beginPath();
canvas1.moveTo(90, 205);
canvas1.lineTo(90, 125);
canvas1.lineTo(10, 205);
canvas1.closePath();
canvas1.stroke();
}
```

3. canvas.html 파일을 저장한다.

역시 파이어폭스를 통해 이 예제 파일을 실행해 보면 그림 2.3과 같이 세 개의 삼각형이 겹쳐진 도형이 나타나는 것을 확인할 수 있다.

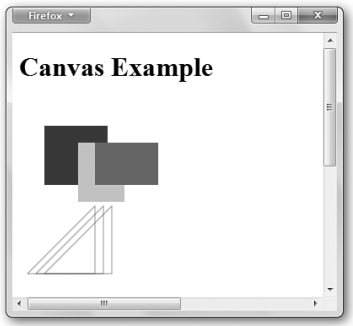

그림 2.3 여러 개의 삼각형 그리기

선 도형 채우기

선으로 그린 도형 역시 내부를 여러분이 원하는 색으로 채울 수 있다. 그 예로 이번에는 삼각형 내부를 초록색으로 채우는 예제 코드를 작성해 보자.

이번 예제에서는 이전 단락에서 예제 코드로 작성했던 코드와 유사한 형태로 beginPath(), moveTo(), lineTo()와 closePath()를 사용해 삼각형을 그릴 것이다. 그러나 모든 코드를 작성한 후 마지막으로 삼각형을 화면에 표시할 때는 앞에서 사용했던 stroke() 함수 대신 fill( ) 함수를 사용할 것이다.

fill() 함수는 여러분이 fillStyle 애트리뷰트로 지정한 색을 지금 그리고 있는 도형의 내부 색으로 채운다. 예를 들어, 밝은 초록색으로 도형의 내부 색을 채우고 싶을 경우는 다음과 같은

코드를 사용하면 된다.

- canvas1.fillStyle = "rgba(0, 200, 0, 0.5)";

그럼 이제 초록색으로 내부 전체를 채운 삼각형을 어떻게 그릴 수 있는지 한번 살펴보자.

1. 메모장과 같은 텍스트 편집기를 사용해 canvas.html 파일을 연다.
2. 그 다음, 초록색으로 채워진 삼각형을 그리기 위해 다음과 같은 코드를 파일에 추가한다.

```
<script type="text/javascript">
  function loader()
  {
  var canvas = document.getElementById("canvas");
  var canvas1 = canvas.getContext("2d");
    .
    .
    .
  //Filled triangle
  canvas1.fillStyle = "rgba(0, 200, 0, 0.5)";
  canvas1.beginPath();
  canvas1.moveTo(225, 25);
  canvas1.lineTo(305, 25);
  canvas1.lineTo(225, 105);
  canvas1.closePath();
  canvas1.fill();
  }
</script>
```

3. canvas.html 파일을 저장한다.

저장한 파일을 실행해 보면 그림 2.4와 같이 초록색으로 채워진 삼각형이 화면에 그려진 것을 볼 수 있다.

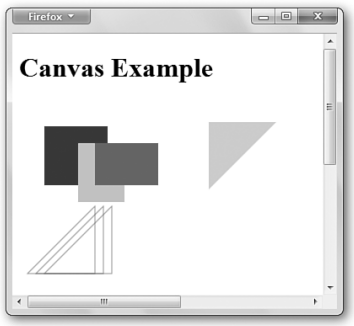

그림 2.4 내부가 채워진 삼각형 그리기

곡선으로 그림 그리기

lineTo() 함수를 사용하면 다양한 형태의 선을 그릴 수 있다. 그러나 부드러운 곡선 처리를 위해서는 bezierCurveTo() 함수를 사용하는 것이 훨씬 더 효과적이다.

- bezierCurveTo(float cp1x, float cp1y, float cp2x, float cp2y, floatx, float y);

그럼 이번에는 부드러운 곡선으로 하트 모양을 그리는 예제 코드에 대해 살펴보자.

1. 메모장과 같은 텍스트 편집기를 사용해 canvas.html 파일을 연다.
2. 내부가 채워진 하트 모양을 그리기 위해 다음과 같은 코드를 추가한다.

```
<script type="text/javascript">
  function loader()
  {
  var canvas = document.getElementById("canvas");
  var canvas1 = canvas.getContext("2d");
    .
    .
    .
  // Heart
  canvas1.fillStyle = "rgba(200, 0, 0, 0.5)";
  canvas1.beginPath();
  canvas1.moveTo(75, 250);
  canvas1.bezierCurveTo(75, 247, 70, 235, 50, 235);
  canvas1.bezierCurveTo(20, 235, 20, 272.5, 20, 272);
  canvas1.bezierCurveTo(20, 290, 40, 312, 75, 330);
  canvas1.bezierCurveTo(110, 312, 130, 290, 130, 272);
  canvas1.bezierCurveTo(130, 272.5, 130, 235, 100, 235);
  canvas1.bezierCurveTo(85, 235, 75, 247, 75, 250);
  canvas1.closePath();
  canvas1.fill();
  }
</script>
```

3. canvas.html 파일을 저장한다.

파일을 브라우저로 실행해 보면 아래에 보이는 그림 2.5와 같이 붉은색의 하트 모양을 볼 수 있다.

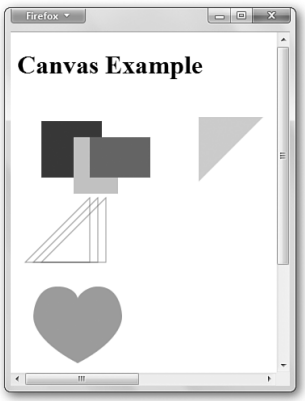

그림 2.5 붉은 색의 하트 모양 그리기

이차 곡선 그리기

이전 예제 코드에서는 부드러운 곡선을 이용해서 하트 모양을 그렸다. 곡선을 이용한 도형 그리기는 이 방법 이외도 quadraticCurveTo( ) 함수를 사용해 이차 곡선을 활용한 도형을 그릴 수도 있다.

- quadraticCurveTo(float cpx, float cpy, float x, float y);

그럼 이번에는 앞서 말한 이차 곡선을 사용한 도형을 그려보자.

1. 메모장과 같은 텍스트 편집기를 사용해 canvas.html 파일을 연다.
2. 이차 곡선으로 도형을 그리기 위해서 다음과 같은 코드를 추가한다.

```
<script type="text/javascript">
  function loader()
  {
  var canvas = document.getElementById("canvas");
  var canvas1 = canvas.getContext("2d");
```

```
    .
    .
    .
  //Quadratic curves
  canvas1.strokeStyle = "rgba(0, 0, 0, 1)";
  canvas1.beginPath();
  canvas1.moveTo(275, 125);
  canvas1.quadraticCurveTo(225, 125, 225, 162);
  canvas1.quadraticCurveTo(260, 200, 265, 200);
  canvas1.quadraticCurveTo(325, 200, 325, 162);
  canvas1.quadraticCurveTo(325, 125, 275, 125);
  canvas1.closePath();
  canvas1.stroke();
  }
</script>
```

3. canvas.html 파일을 저장한다.

저장한 파일을 브라우저로 열어 보면 그림 2.6과 동일한 형태의 도형이 나타날 것이다.

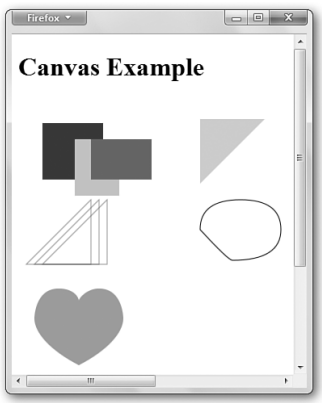

그림 2.6 이차 곡선을 이용한 도형 그리기

둥근 모양 그리기

다음과 같은 함수를 사용하면 캔버스에 둥근 모양의 그림도 그릴 수 있다.

- arc(float x, float y, float radius, float startAngle, float endAngle, boolean anticlockwise);

그럼 다음과 같은 단계를 통해 둥근 모양의 도형을 그려보자.

1. 메모장과 같은 텍스트 편집기를 사용해 canvas.html 파일을 연다.
2. 둥근 모형을 생성하기 위해서 다음과 같은 코드를 파일에 추가한다(여기서 주의해야 할 점은 파이(pi) 값을 가져오기 위해서 자바스크립트의 상수인 Math.PI를 사용하는 부분이다.)

```
<script type="text/javascript">
  function loader()
```

```
  {
  var canvas = document.getElementById("canvas");
  var canvas1 = canvas.getContext("2d");
    .
    .
    .
  // Arcs
  canvas1.beginPath();
  canvas1.arc(275, 275, 50, 0, Math.PI * 2, true);

  canvas1.moveTo(310, 275);
  canvas1.arc(275, 275, 35, 0, 0.75 * Math.PI, false);

  canvas1.moveTo(300, 255);
  canvas1.arc(265, 255, 35, 0, 0.5 * Math.PI, false);

  canvas1.moveTo(280, 255);
  canvas1.arc(245, 255, 35, 0, 0.2 * Math.PI, false);
  canvas1.closePath();
  canvas1.stroke();
  }
</script>
```

3. canvas.html 파일을 저장한다.

역시 파일을 브라우저로 실행해 보면 다음의 그림 2.7과 같이 둥근 모양의 그림이 그려진 것을 확인할 수 있다.

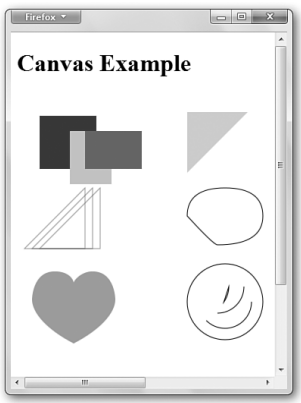

그림 2.7 둥근 모양의 도형 그리기

문자열 그리기

Canvas 컨트롤을 사용해서 문자열을 그릴 수도 있다. 그러기 위해서는 먼저 원하는 글씨체를 지정해야 하는데, 이때는 다음에 보이는 간단한 예제 코드와 같이 Canvas 컨트롤의 font 애트리뷰트를 사용해서 글씨체를 지정하면 된다.

- canvas1.font = "italic 40px sans-serif";

이 코드는 자바스크립트를 사용해서 기본 글씨체로 40픽셀 글씨 높이의 san-serif 이탤릭체를 지정한다(만일 이탤릭 글씨체를 원치 않는다면, 'italic'을 생략하면 된다).

여러분이 원하는 글씨체를 지정한 후에 strokeText( ) 함수를 사용해서 문자열을 그리면 되는데, 이때는 그리고자 하는 문자열과 문자열을 그릴 위치를 함수의 매개변수로 전달해야 한다. 그럼 실제로 아래와 같은 단계를 통해 직접 문자열을 한번 그려보자.

1. 메모장과 같은 텍스트 편집기를 사용해 canvas.html 파일을 연다.
2. 문자열을 그리기 위해 아래의 코드를 추가한다.

```
<script type="text/javascript">
  function loader()
  {
  var canvas = document.getElementById("canvas");
  var canvas1 = canvas.getContext("2d");
    .
    .
    .
  canvas1.font = "italic 40px sans-serif";
  canvas1.strokeText("Hello!", 50, 400);
  }
</script>
```

3. canvas.html 파일을 저장한다.

파일을 저장한 후 브라우저를 통해 파일을 실행해 보면 그림 2.8과 같이 문자열이 화면에 그려졌음을 확인할 수 있다.

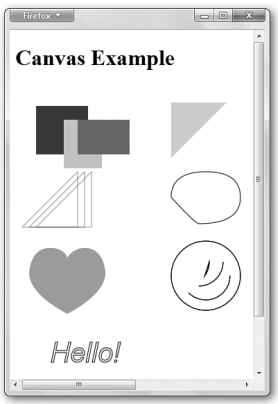

그림 2.8 문자열 그리기

canvas.html 예제 코드

지금까지 이 장을 통해서 설명하고자 했던 Canvas 컨트롤의 기능을 모두 살펴봤다. 다음에 나오는 코드는 이 장에서 설명한 전체 코드를 작성해 놓은 것이다. 이 장의 내용을 확인할 때 참고하기 바란다.

```
<!DOCTYPE html>
<html>
  <head>
    <title>
      Canvas Example
    </title>

    <script type="text/javascript">
      function loader()
      {
      var canvas = document.getElementById("canvas");
      var canvas1 = canvas.getContext("2d");

      // Rectangles
      canvas1.fillStyle = "rgba(0, 0, 200, 1)";
      canvas1.fillRect(30, 30, 75, 70);

      canvas1.fillStyle = "rgba(200, 200, 0, 1)";
      canvas1.fillRect(70, 50, 55, 70);

      canvas1.fillStyle = "rgba(200, 0, 0, 1)";
      canvas1.fillRect(90, 50, 75, 50);

      //Filled triangle
      canvas1.fillStyle = "rgba(0, 200, 0, 0.5)";
      canvas1.beginPath();
      canvas1.moveTo(225, 25);
      canvas1.lineTo(305, 25);
      canvas1.lineTo(225, 105);
      canvas1.closePath();
      canvas1.fill();

      // Stroked triangles
      canvas1.beginPath();
      canvas1.strokeStyle = "rgba(200, 0, 0, 0.5)";
      canvas1.moveTo(110, 205);
      canvas1.lineTo(110, 125);
      canvas1.lineTo(30, 205);
      canvas1.closePath();
      canvas1.stroke();
```

```
canvas1.beginPath();
canvas1.moveTo(100, 205);
canvas1.lineTo(100, 125);
canvas1.lineTo(20, 205);
canvas1.closePath();
canvas1.stroke();

canvas1.beginPath();
canvas1.moveTo(90, 205);
canvas1.lineTo(90, 125);
canvas1.lineTo(10, 205);
canvas1.closePath();
canvas1.stroke();

// Heart
canvas1.fillStyle = "rgba(200, 0, 0, 0.5)";
canvas1.beginPath();
canvas1.moveTo(75, 250);
canvas1.bezierCurveTo(75, 247, 70, 235, 50, 235);
canvas1.bezierCurveTo(20, 235, 20, 272.5, 20, 272);
canvas1.bezierCurveTo(20, 290, 40, 312, 75, 330);
canvas1.bezierCurveTo(110, 312, 130, 290, 130, 272);
canvas1.bezierCurveTo(130, 272.5, 130, 235, 100, 235);
canvas1.bezierCurveTo(85, 235, 75, 247, 75, 250);
canvas1.closePath();
canvas1.fill();

//Quadratic curves
canvas1.strokeStyle = "rgba(0, 0, 0, 1)";
canvas1.beginPath();
canvas1.moveTo(275, 125);
canvas1.quadraticCurveTo(225, 125, 225, 162);
canvas1.quadraticCurveTo(260, 200, 265, 200);
canvas1.quadraticCurveTo(325, 200, 325, 162);
canvas1.quadraticCurveTo(325, 125, 275, 125);
canvas1.closePath();
canvas1.stroke();

// Arcs
canvas1.beginPath();
canvas1.arc(275, 275, 50, 0, Math.PI * 2, true);
```

```
      canvas1.moveTo(310, 275);
      canvas1.arc(275, 275, 35, 0, 0.75 * Math.PI, false);

      canvas1.moveTo(300, 255);
      canvas1.arc(265, 255, 35, 0, 0.5 * Math.PI, false);

      canvas1.moveTo(280, 255);
      canvas1.arc(245, 255, 35, 0, 0.2 * Math.PI, false);
      canvas1.closePath();

      canvas1.stroke();

      canvas1.font = 'italic 40px sans-serif';
      canvas1.strokeText("Hello!", 50, 400);
      }
    </script>
  </head>

  <body onload="loader()">
    <h1>Canvas Example</h1>
    <canvas id="canvas" width="600" height="500">
    </canvas>

  </body>
</html>
```