What I have added

HTML:

```
<!--meta tags for responsiveness-->
  <meta name="viewport" content="width=device-width,initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">
  <meta name="HandheldFriendly" content="true">
  <meta charset="utf-8">
```

```
<div class="settings">
    <p>Choose an effect:</p>

    <label for="none">none</label>
    <input type="radio" id="none" name="effect">

    <label for="source-atop">source-atop</label>
    <input type="radio" id="source-atop" name="effect">

    <label for="multiply">multiply</label>
    <input type="radio" id="multiply" name="effect">

    <label for="destination-over">destination-over</label>
    <input type="radio" id="destination-over" name="effect">

    <label for="destination-in">destination-in</label>
    <input type="radio" id="destination-in" name="effect">

    <label for="destination-out">destination-out</label>
    <input type="radio" id="destination-out" name="effect">

    <label for="destination-atop">destination-atop</label>
    <input type="radio" id="destination-atop" name="effect">

    <label for="lighter">lighter</label>
    <input type="radio" id="lighter" name="effect">

    <label for="copy">copy</label>
    <input type="radio" id="copy" name="effect">

    <label for="xor">xor</label>
    <input type="radio" id="xor" name="effect">

    <label for="screen">screen</label>
    <input type="radio" id="screen" name="effect">

    <label for="overlay">overlay</label>
```

```
    <input type="radio" id="overlay" name="effect">

    <label for="darken">darken</label>
    <input type="radio" id="darken" name="effect">

    <label for="lighten">lighten</label>
    <input type="radio" id="lighten" name="effect">

    <label for="color-dodge">color-dodge</label>
    <input type="radio" id="color-dodge" name="effect">

    <label for="color-burn">color-burn</label>
    <input type="radio" id="color-burn" name="effect">

    <label for="hard-light">hard-light</label>
    <input type="radio" id="hard-light" name="effect">

    <label for="soft-light">soft-light</label>
    <input type="radio" id="soft-light" name="effect">

    <label for="difference">difference</label>
    <input type="radio" id="difference" name="effect">

    <label for="exclusion">exclusion</label>
    <input type="radio" id="exclusion" name="effect">

    <label for="hue">hue</label>
    <input type="radio" id="hue" name="effect">

    <label for="saturation">saturation</label>
    <input type="radio" id="saturation" name="effect">

    <label for="color">color</label>
    <input type="radio" id="color" name="effect">

    <label for="luminosity">luminosity</label>
    <input type="radio" id="luminosity" name="effect">

    <label for="line-width">Line Width:</label>
    <input type="number" id="line-width" value="50" min="1" max="100">

    <label for="auto-width">Auto Width</label>
    <input type="checkbox" id="auto-width">

    <!-- Clear Canvas Button -->
    <button id="clear-canvas">Clear Canvas</button>
  </div>

  <canvas id="draw"></canvas>
```

CSS:

```
html,
    body {
      margin: 0;
      width: 100%;
      height: 100%;
    }

    html {
      max-height: -webkit-fill-available;
      overflow-y: hidden;
    }

    canvas {
      width: -webkit-fill-available;
      height: -webkit-fill-available;
    }

    .settings {
      position: absolute;
      background-color: black;
      color: aliceblue;
      padding: 1%;
    }
```

Script:

```
const radioButtons = document.querySelectorAll('input[name="effect"]');
    radioButtons.forEach(radio => {
      radio.addEventListener('change', function () {
        switch (this.id) {
          case 'none':
            ctx.globalCompositeOperation = 'source-over';
            break;
          case 'source-atop':
            ctx.globalCompositeOperation = 'source-atop';
            break;
          case 'multiply':
            ctx.globalCompositeOperation = 'multiply';
            break;
          case 'destination-over':
            ctx.globalCompositeOperation = 'destination-over';
            break;
          case 'destination-in':
            ctx.globalCompositeOperation = 'destination-in';
            break;
```

```
          case 'destination-out':
            ctx.globalCompositeOperation = 'destination-out';
            break;
          case 'destination-atop':
            ctx.globalCompositeOperation = 'destination-atop';
            break;
          case 'lighter':
            ctx.globalCompositeOperation = 'lighter';
            break;
          case 'copy':
            ctx.globalCompositeOperation = 'copy';
            break;
          case 'xor':
            ctx.globalCompositeOperation = 'xor';
            break;
          case 'screen':
            ctx.globalCompositeOperation = 'screen';
            break;
          case 'overlay':
            ctx.globalCompositeOperation = 'overlay';
            break;
          case 'darken':
            ctx.globalCompositeOperation = 'darken';
            break;
          case 'lighten':
            ctx.globalCompositeOperation = 'lighten';
            break;
          case 'color-dodge':
            ctx.globalCompositeOperation = 'color-dodge';
            break;
          case 'color-burn':
            ctx.globalCompositeOperation = 'color-burn';
            break;
          case 'hard-light':
            ctx.globalCompositeOperation = 'hard-light';
            break;
          case 'soft-light':
            ctx.globalCompositeOperation = 'soft-light';
            break;
          case 'difference':
            ctx.globalCompositeOperation = 'difference';
            break;
          case 'exclusion':
            ctx.globalCompositeOperation = 'exclusion';
            break;
          case 'hue':
            ctx.globalCompositeOperation = 'hue';
            break;
          case 'saturation':
```

```
          ctx.globalCompositeOperation = 'saturation';
          break;
        case 'color':
          ctx.globalCompositeOperation = 'color';
          break;
        case 'luminosity':
          ctx.globalCompositeOperation = 'luminosity';
          break;

        default:

          break;
      }
    });
  });

  // Add event listener to adjust line width
  const lineWidthInput = document.getElementById('line-width');
  lineWidthInput.addEventListener('input', function () {
    ctx.lineWidth = parseInt(this.value);
  });

  // Add event listener for "Auto Width" checkbox
  const autoWidthCheckbox = document.getElementById('auto-width');
  autoWidthCheckbox.addEventListener('change', function () {
    autoWidth = this.checked;
  });

  // Clear Canvas Button
  const clearCanvasButton = document.getElementById('clear-canvas');
  clearCanvasButton.addEventListener('click', function () {
    ctx.clearRect(0, 0, canvas.width, canvas.height);
  });
```

As an extra feature, I have made a responsive design to the web page by including meta tags for viewport settings. This ensures that the content scales appropriately on various devices.

Additionally, I improved the overall layout by setting the width and height of the html and body elements to 100%, and I used the -webkit-fill-available property for the canvas to make it occupy the entire available space within its container.

Furthermore, I introduced a settings panel containing radio buttons for selecting different global composite operation effects, an input field to adjust the line width of the drawing, and a checkbox to enable or disable the auto-width feature. These settings provide users with more control over the drawing appearance.

Finally, a "Clear Canvas" button was added, allowing users to easily reset the canvas and start over. These enhancements aim to improve the user experience and offer more customization options for the drawing application.

Update

I have addressed feedback from my teachers by improving the clarity of the settings panel. To eliminate confusion, I removed checkboxes with similar effects. Additionally, I repositioned the settings panel to the left for better organization. Each option now includes a small explanation icon, and hovering over it reveals a tooltip with a brief description.

Furthermore, I added a footer containing a copyright statement and a link to the website that provided the script names for the checkbox functions. These enhancements aim to improve both the usability and understanding of the HTML5 Canvas drawing application.

I have created the resizeCanvas() function to invoke the canvas resize function initially to set up the canvas correctly when resizing the screenwidth.

HTML:

```
<div class="settings">
    <p>Choose an effect:</p>

    <div class="option">
      <div class="circle-button">i
        <span class="tooltip">This is the default mode.</span>
      </div>
      <label for="none">none</label>
      <input type="radio" id="none" name="effect" checked>
    </div>

    <div class="option">
      <div class="circle-button">i
        <span class="tooltip">The new shape is only drawn where it overlaps
the existing canvas content.</span>
      </div>
      <label for="source-atop">source-atop</label>
      <input type="radio" id="source-atop" name="effect">
    </div>

    <div class="option">
      <div class="circle-button">i
        <span class="tooltip">
          The pixels of the top layer are multiplied with the corresponding
pixels of the bottom
          layer. A darker picture is the result.
        </span>
```

```
      </div>
      <label for="multiply">multiply</label>
      <input type="radio" id="multiply" name="effect">
    </div>

    <div class="option">
      <div class="circle-button">i
        <span class="tooltip">New shapes are drawn behind the existing canvas
content.</span>
      </div>
      <label for="destination-over">destination-over</label>
      <input type="radio" id="destination-over" name="effect">
    </div>

    <div class="option">
      <div class="circle-button">i
        <span class="tooltip">
          The existing canvas content is kept where both the new shape and
existing canvas content
          overlap. Everything else is made transparent.
        </span>
      </div>
      <label for="destination-in">destination-in</label>
      <input type="radio" id="destination-in" name="effect">
    </div>

    <div class="option">
      <div class="circle-button">i
        <span class="tooltip">The existing content is kept where it doesn't
overlap the new shape.</span>
      </div>
      <label for="destination-out">destination-out</label>
      <input type="radio" id="destination-out" name="effect">
    </div>

    <div class="option">
      <div class="circle-button">i
        <span class="tooltip">The new shape is only drawn where it overlaps
the existing canvas content.</span>
      </div>
      <label for="destination-atop">destination-atop</label>
      <input type="radio" id="destination-atop" name="effect">
    </div>

    <div class="option">
      <div class="circle-button">i
        <span class="tooltip">Shapes are made transparent where both overlap
and drawn normal everywhere else.</span>
      </div>
```

```
      <label for="xor">xor</label>
      <input type="radio" id="xor" name="effect">
    </div>

    <div class="option">
      <div class="circle-button">i
        <span class="tooltip">Retains the darkest pixels of both
layers.</span>
      </div>
      <label for="darken">darken</label>
      <input type="radio" id="darken" name="effect">
    </div>

    <div class="option">
      <div class="circle-button">i
        <span class="tooltip">Retains the lightest pixels of both
layers.</span>
      </div>
      <label for="lighten">lighten</label>
      <input type="radio" id="lighten" name="effect">
    </div>

    <div class="option">
      <div class="circle-button">i
        <span class="tooltip">
          Subtracts the bottom layer from the top layer — or the other way
round — to always get a
          positive value.
        </span>
      </div>
      <label for="difference">difference</label>
      <input type="radio" id="difference" name="effect">
    </div>

    <div class="option">
      <div class="circle-button">i
        <span class="tooltip">Like difference, but with lower contrast.</span>
      </div>
      <label for="exclusion">exclusion</label>
      <input type="radio" id="exclusion" name="effect">
    </div>

    <div class="option">
      <div class="circle-button">i
        <span class="tooltip">
          Preserves the luma of the bottom layer, while adopting the hue and
chroma of the top
          layer.
        </span>
```

```
      </div>
      <label for="color">color</label>
      <input type="radio" id="color" name="effect">
    </div>

    <div class="option">
      <div class="circle-button">i
        <span class="tooltip">Change the width of the tool.</span>
      </div>
      <label for="line-width">Line Width:</label>
      <input type="number" id="line-width" value="50" min="1" max="100">
    </div>

    <div class="option">
      <div class="circle-button">i
        <span class="tooltip">Automatically changes th widt of the tool up to
100px and back.</span>
      </div>
      <label for="auto-width">Auto Width</label>
      <input type="checkbox" id="auto-width">
    </div>

    <!-- Clear Canvas Button -->
    <button id="clear-canvas">Clear Canvas</button>
  </div>
  <footer>
    © 2024 Mozilla Foundation. All rights reserved. The content on this page
is licensed under the Creative Commons
    Attribution-ShareAlike 2.5 Generic license or any later version. See the
'License' section in the provided link:
    <br>
    <a href="https://developer.mozilla.org/en-
US/docs/Web/API/CanvasRenderingContext2D/globalCompositeOperation"
      target="_blank">https://developer.mozilla.org/en-
US/docs/Web/API/CanvasRenderingContext2D/globalCompositeOperation</a>
  </footer>

  <canvas id="draw"></canvas>
```

CSS:

```
.settings label {
      display: block;
      margin-bottom: 5px;
    }

    .settings input,
    .settings button {
      margin-bottom: 10px;
    }

    .option {
      display: flex;
      align-items: flex-start;
      margin-bottom: 5px;
      justify-content: space-between;
    }

    .option label {
      margin: 0 10px;
    }

    .circle-button {
      display: flex;
      width: 20px;
      height: 20px;
      background-color: grey;
      border-radius: 50%;
      color: white;
      text-align: center;
      line-height: 40px;
      font-size: 13px;
      font-family: Arial, sans-serif;
      align-items: center;
      justify-content: center;
      position: relative;
      cursor: pointer
    }

    .circle-button .tooltip {
      visibility: hidden;
      min-width: 9rem;
      background-color: grey;
      color: #fff;
      text-align: left;
      padding: 1rem;
      border-radius: 6px;
      position: absolute;
```

```
      z-index: 1;
      top: 100%;
      left: 1.3rem;
      line-height: normal;
    }

    .circle-button:hover .tooltip {
      visibility: visible;
    }

    footer {
      background-color: grey;
      position: fixed;
      bottom: 0;
      padding: 5px 2rem 5px 15rem;
      color: aliceblue;
      font-size: 15px;
      text-align: left;
    }
```

Script:

```
const radioButtons = document.querySelectorAll('input[name="effect"]');
    radioButtons.forEach(radio => {
      radio.addEventListener('change', function () {
        switch (this.id) {
          case 'none':
            ctx.globalCompositeOperation = 'source-over';
            break;
          case 'source-atop':
            ctx.globalCompositeOperation = 'source-atop';
            break;
          case 'multiply':
            ctx.globalCompositeOperation = 'multiply';
            break;
          case 'destination-over':
            ctx.globalCompositeOperation = 'destination-over';
            break;
          case 'destination-in':
            ctx.globalCompositeOperation = 'destination-in';
            break;
          case 'destination-out':
            ctx.globalCompositeOperation = 'destination-out';
            break;
          case 'destination-atop':
            ctx.globalCompositeOperation = 'destination-atop';
            break;
          case 'xor':
            ctx.globalCompositeOperation = 'xor';
```

```
          break;
        case 'darken':
          ctx.globalCompositeOperation = 'darken';
          break;
        case 'lighten':
          ctx.globalCompositeOperation = 'lighten';
          break;
        case 'soft-light':
          ctx.globalCompositeOperation = 'soft-light';
          break;
        case 'difference':
          ctx.globalCompositeOperation = 'difference';
          break;
        case 'exclusion':
          ctx.globalCompositeOperation = 'exclusion';
          break;
        case 'color':
          ctx.globalCompositeOperation = 'color';
          break;

        default:

          break;
      }
    });
  });

  // Add event listener to adjust line width
  const lineWidthInput = document.getElementById('line-width');
  lineWidthInput.addEventListener('input', function () {
    ctx.lineWidth = parseInt(this.value);
  });

  // Add event listener for "Auto Width" checkbox
  const autoWidthCheckbox = document.getElementById('auto-width');
  autoWidthCheckbox.addEventListener('change', function () {
    autoWidth = this.checked;
  });

  // Clear Canvas Button
  const clearCanvasButton = document.getElementById('clear-canvas');
  clearCanvasButton.addEventListener('click', function () {
    ctx.clearRect(0, 0, canvas.width, canvas.height);
  });

  function resizeCanvas() {
    canvas.width = window.innerWidth;
    canvas.height = window.innerHeight;
    ctx.lineWidth = 50;
```

```
    ctx.lineJoin = 'round';
    ctx.lineCap = 'round';
    ctx.strokeStyle = '#BADA55';
    ctx.globalCompositeOperation = 'source-over';
  }

  // Initial resize to set up canvas correctly
  resizeCanvas();

  // Event listener for window resize
  window.addEventListener('resize', resizeCanvas);
```

Overall, the changes have significantly improved the user interface, provided more customization options, and enhanced the overall user experience of the HTML5 Canvas drawing application.