HTML CODE:

```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <title>Bubble Sort Visualizer (Speed Control)</title>
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <link rel="stylesheet" href="styles.css" />
</head>
<body>
  <div class="container">
    <h1>Bubble Sort Visualizer</h1>
    <div class="controls">
      <label for="array-input">Enter array (comma or space separated):</label>
      <input id="array-input" placeholder="e.g., 5,2,9,1,5,6" />

      <div class="row">
        <div class="speed-control">
          <label for="speed-slider">Animation speed:</label>
          <div class="slider-row">
            <input id="speed-slider" type="range" min="10" max="1000" value="200" />
            <span id="speed-display">200 ms</span>
          </div>
          <div class="hint">Lower is faster.</div>
        </div>
      </div>

      <div class="buttons">
        <button id="parse-btn">Load Array</button>
        <button id="shuffle-btn">Shuffle</button>
        <button id="start-btn">Start</button>
        <button id="pause-btn" disabled>Pause</button>
        <button id="reset-btn">Reset</button>
      </div>
      <div id="status" class="status">Idle</div>
    </div>

    <div id="visualization" class="visualization"></div>
    <div class="legend">
      <span class="legend-item compare">Comparing</span>
      <span class="legend-item swap">Swapping</span>
      <span class="legend-item sorted">Sorted</span>
    </div>
  </div>

  <script src="script.js"></script>
</body>
</html>
```

CSS CODE

```
* {
  box-sizing: border-box;
}
body {
  margin: 0;
  font-family: system-ui,-apple-system,BlinkMacSystemFont,sans-serif;
  background: #1f2937;
  color: #f0f5f9;
}
.container {
  max-width: 960px;
  margin: 0 auto;
  padding: 16px;
}
h1 {
  margin-top: 0;
  font-size: 1.9rem;
  text-align: center;
}
.controls {
  background: #111f38;
  padding: 16px;
  border-radius: 12px;
  display: grid;
  gap: 12px;
}
.controls input[type="text"],
.controls input[type="number"],
.controls input[type="range"] {
  width: 100%;
  padding: 6px 10px;
  border-radius: 6px;
  border: none;
  font-size: 1rem;
}
.row {
  display: flex;
  gap: 10px;
  align-items: flex-start;
  flex-wrap: wrap;
}
.speed-control {
  flex: 1 1 100%;
}
.slider-row {
  display: flex;
  gap: 8px;
  align-items: center;
}
.hint {
  font-size: 0.65rem;
  margin-top: 4px;
  color: #c0cddb;
}
.buttons {
  display: flex;
  gap: 10px;
  flex-wrap: wrap;
}
button {
  padding: 10px 16px;
  border: none;
  border-radius: 8px;
  cursor: pointer;
  background: #2563eb;
  color: white;
  font-weight: 600;
  transition: filter .2s;
}
button:hover { filter: brightness(1.1); }
button:disabled {
  opacity: .5;
  cursor: not-allowed;
```

```
}
.status {
  margin-top: 8px;
  font-size: .9rem;
  background: rgba(255,255,255,.05);
  padding: 8px 12px;
  border-radius: 6px;
  display: inline-block;
}
.visualization {
  margin-top: 24px;
  position: relative;
  height: 360px;
  background: #0f1b33;
  border-radius: 12px;
  padding: 16px;
  display: flex;
  align-items: flex-end;
  gap: 6px;
  overflow: hidden;
}
.bar {
  flex: 1 1 auto;
  position: relative;
  display: flex;
  align-items: flex-end;
  justify-content: center;
  border-radius: 4px;
  transition: transform .3s ease;
}
.bar-inner {
  width: 100%;
  border-radius: 4px;
  position: relative;
  display: flex;
  align-items: flex-end;
  justify-content: center;
  font-size: 0.65rem;
  font-weight: 600;
  padding-bottom: 4px;
  color: #fff;
}
.compare {
  background: #f59e0b !important;
}
.swap {
  background: #ef4444 !important;
}
.sorted {
  background: #10b981 !important;
}
.legend {
  margin-top: 16px;
  display: flex;
  gap: 16px;
  flex-wrap: wrap;
}
.legend-item {
  padding: 6px 12px;
  border-radius: 6px;
  font-size: 0.75rem;
  display: inline-flex;
  align-items: center;
  gap: 6px;
}
.legend-item.compare { background: #f59e0b; }
.legend-item.swap { background: #ef4444; }
.legend-item.sorted { background: #10b981; }
@media (max-width: 800px) {
  .visualization { height: 260px; }
}
```

JAVA SCRIPT

```
// Helper
const delay = ms => new Promise(r => setTimeout(r, ms));

let array = [];
let bars = [];
let sorting = false;
let paused = false;
let stopRequested = false;

const viz = document.getElementById('visualization');
const statusEl = document.getElementById('status');
const inputEl = document.getElementById('array-input');
const speedSlider = document.getElementById('speed-slider');
const speedDisplay = document.getElementById('speed-display');
const startBtn = document.getElementById('start-btn');
const pauseBtn = document.getElementById('pause-btn');
const parseBtn = document.getElementById('parse-btn');
const shuffleBtn = document.getElementById('shuffle-btn');
const resetBtn = document.getElementById('reset-btn');
```

```
function setStatus(txt) {
  statusEl.textContent = txt;
}

function buildBars() {
  viz.innerHTML = '';
  bars = [];
  if (!Array.isArray(array) || array.length === 0) return;
  const maxVal = Math.max(...array.map(v => Math.abs(v))) || 1;
  array.forEach((val, i) => {
    const bar = document.createElement('div');
    bar.className = 'bar';
    const inner = document.createElement('div');
    inner.className = 'bar-inner';
    inner.textContent = val;
    const heightPercent = Math.abs(val) / maxVal;
    inner.style.height = `${Math.max(8, heightPercent * 100)}%`;
    inner.style.background = '#3b82f6';
    bar.appendChild(inner);
    viz.appendChild(bar);
    bars.push({ barEl: bar, innerEl: inner, value: val });
  });
}

function parseInput() {
  const raw = inputEl.value.trim();
  if (!raw) return [];
  const parts = raw.split(/[\s,]+/);
  const nums = parts.map(p => {
    const n = parseFloat(p);
    return isNaN(n) ? null : n;
  }).filter(v => v !== null);
  return nums;
}

function disableControls(state) {
  startBtn.disabled = state;
  parseBtn.disabled = state;
  shuffleBtn.disabled = state;
  resetBtn.disabled = state;
}

function getCurrentSpeed() {
  // slider value is milliseconds per comparison/swap
  return Math.max(10, parseInt(speedSlider.value) || 200);
}

async function bubbleSort() {
  if (sorting) return;
  sorting = true;
  paused = false;
  stopRequested = false;
  pauseBtn.textContent = 'Pause';
  pauseBtn.disabled = false;
  disableControls(true);
  setStatus('Sorting...');
  const n = bars.length;

  for (let i = 0; i < n - 1; i++) {
    let swapped = false;
    for (let j = 0; j < n - i - 1; j++) {
```

```
    if (stopRequested) {
      setStatus('Stopped');
      cleanupAfterSort();
      return;
    }
    // handle pause
    while (paused) {
      setStatus('Paused');
      await delay(100);
    }
    // comparison
    highlight(j, j + 1, 'compare');
    setStatus(`Comparing indices ${j} & ${j + 1}`);
    await delay(getCurrentSpeed());

    if (bars[j].value > bars[j + 1].value) {
      highlight(j, j + 1, 'swap');
      setStatus(`Swapping ${bars[j].value} and ${bars[j + 1].value}`);
      await animateSwap(j, j + 1);
      [bars[j], bars[j + 1]] = [bars[j + 1], bars[j]];
      swapped = true;
    }

    unhighlight(j, j + 1);
    await delay(60);
  }
  markSorted(n - i - 1);
  if (!swapped) break;
}

for (let k = 0; k < n; k++) markSorted(k);
```

```
  setStatus('Sorted ✔');
  cleanupAfterSort();
}

function cleanupAfterSort() {
  sorting = false;
  pauseBtn.disabled = true;
  disableControls(false);
}

function highlight(i, j, cls) {
  [bars[i].innerEl, bars[j].innerEl].forEach(el => {
    el.classList.remove('compare', 'swap', 'sorted');
    el.classList.add(cls);
    if (cls === 'compare') el.style.background = '#f59e0b';
    if (cls === 'swap') el.style.background = '#ef4444';
  });
}

function unhighlight(i, j) {
  [bars[i].innerEl, bars[j].innerEl].forEach(el => {
    el.classList.remove('compare', 'swap');
    el.style.background = '#2563eb';
    el.style.transform = '';
  });
}

function markSorted(index) {
  const inner = bars[index].innerEl;
  inner.classList.remove('compare', 'swap');
  inner.classList.add('sorted');
  inner.style.background = '#10b981';
}

async function animateSwap(i, j) {
  const a = bars[i];
  const b = bars[j];
  const aHeight = a.innerEl.style.height;
  const bHeight = b.innerEl.style.height;
  const aVal = a.value;
  const bVal = b.value;

  // quick pop
  a.innerEl.style.transform = 'scale(1.1)';
  b.innerEl.style.transform = 'scale(1.1)';
  await delay(getCurrentSpeed() / 2);
  a.innerEl.style.transform = '';
  b.innerEl.style.transform = '';

  // swap
  a.innerEl.textContent = bVal;
  b.innerEl.textContent = aVal;
  a.innerEl.style.height = bHeight;
  b.innerEl.style.height = aHeight;
  a.value = bVal;
  b.value = aVal;
}

function shuffleArray() {
  if (!array || array.length === 0) return;
  for (let i = array.length - 1; i > 0; i--) {
    const j = Math.floor(Math.random() * (i + 1));
    [array[i], array[j]] = [array[j], array[i]];
  }
  buildBars();
  setStatus('Shuffled array.');
}

function resetAll() {
  sorting = false;
  paused = false;
  stopRequested = true;
  disableControls(false);
  array = [];
  inputEl.value = '';
  buildBars();
  setStatus('Reset.');
  pauseBtn.disabled = true;
  pauseBtn.textContent = 'Pause';
}

// Event wiring
parseBtn.addEventListener('click', () => {
  const parsed = parseInput();
  if (parsed.length === 0) {
    setStatus('Invalid or empty array.');
    return;
  }
  array = parsed.slice();
  buildBars();
  setStatus('Array loaded.');
});
```

```
shuffleBtn.addEventListener('click', () => {
  if (array.length === 0) {
    setStatus('Load an array first.');
    return;
  }
  shuffleArray();
});

startBtn.addEventListener('click', async () => {
  if (array.length === 0) {
    setStatus('Load an array first.');
    return;
  }
  if (!sorting) {
    bubbleSort();
  }
});

pauseBtn.addEventListener('click', () => {
  if (!sorting) return;
  paused = !paused;
  pauseBtn.textContent = paused ? 'Resume' : 'Pause';
  if (!paused) setStatus('Resuming...');
});

resetBtn.addEventListener('click', resetAll);

// live speed display
speedSlider.addEventListener('input', () => {
  speedDisplay.textContent = `${speedSlider.value} ms`;
  if (sorting && !paused) {
    setStatus(`Speed adjusted to ${speedSlider.value} ms`);
  }
});

// allow enter to load array
inputEl.addEventListener('keydown', e => {
  if (e.key === 'Enter') parseBtn.click();
});

// init example
window.addEventListener('load', () => {
  inputEl.value = '5,3,8,2,9,1';
  parseBtn.click();
});
```