Using SVG in the Web Browser

Information Visualisation 2024
08 May 2024

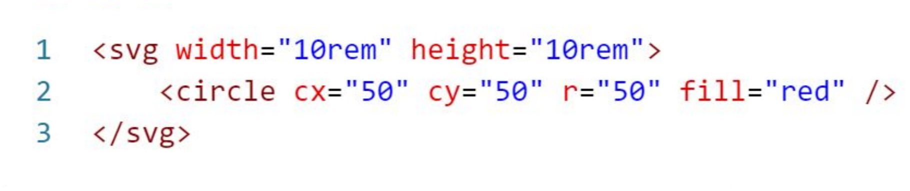
Aumüller Thomas, Heider Martin, Ramadan Abdelrahman

Scalable Vector Graphics (SVG)

XML-based two-dimensional graphics, supporting interactivity, scalability and animation.

Scalable without loss of quality, searchable, indexable, scriptable, and compressible.

Editable with **text editors**, and supported by **most web browsers**.



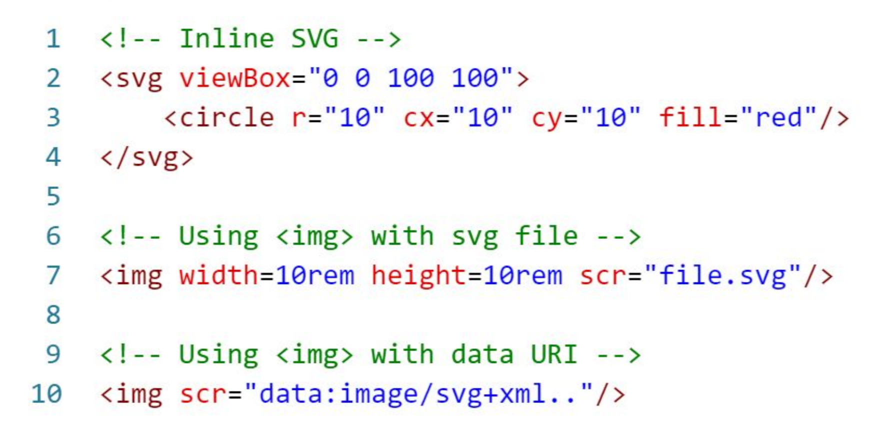
```
<svg width="10rem" height="10rem">
    <circle cx="50" cy="50" r="50" fill="red" />
</svg>
```

Including SVG

- **Inline SVG**
  - Supported in all modern Browsers.
  - Part of the page DOM.
  - Clutters the page.

- **Using `<img>` with svg file**
  - Ease of use.
  - Fallbacks with `alt=` attribute.
  - SVG treated as any other image.

- **Using `<img>` with data URI**
  - Reduced HTTP requests.
  - Elimination of dependencies.
  - Opaqueness.



```html
<!-- Inline SVG -->
<svg viewBox="0 0 100 100">
    <circle r="10" cx="10" cy="10" fill="red"/>
</svg>

<!-- Using <img> with svg file -->
<img width=10rem height=10rem scr="file.svg"/>

<!-- Using <img> with data URI -->
<img scr="data:image/svg+xml.."/>
```

Including SVG

- **CSS background image**
  - Flexibility with CSS styling options.
  - Interactivity via JavaScript event handling.
  - Compatibility issues may arise.

```css
.demo1 {
  width: 10rem;
  height: 10rem;
  background-image: url('sizing_svg.svg');
}
```

- **CSS background with data URI**
  - Improved performance.
  - Additional layer of complexity.

```css
.demo2 {
  width: 10rem;
  height: 10rem;
  background-image: url('data:image/svg+xml..');
}
```

- **SVG-DOM injection**
  - Dynamic and flexible
  - Improved performance.
  - Highest complexity and maintenance.

```html
<script>
// Create SVG element
var svgNS = "http://www.w3.org/2000/svg";
var svg = document.createElementNS(svgNS, "svg");
svg.setAttribute("width", "10rem");
svg.setAttribute("height", "10rem");

// Create SVG circle element
var circle = document.createElementNS(svgNS, "circle");
circle.setAttribute("cx", "100");
circle.setAttribute("cy", "100");
circle.setAttribute("r", "50");
circle.setAttribute("fill", "red");

// Append circle to SVG
svg.appendChild(circle);

// Append SVG to container
var container = document.getElementById("svg-container");
container.appendChild(svg);
</script>
```

Styling SVGs

1. Inline Styles

```
style="fill: red; stroke: blue; stroke-width: 2;
```

2. Internal Stylesheets

```
<style>
    rect {
      fill: rgb(0, 128, 0);
      stroke: yellow;
      stroke-width: 4;
    }
</style>
```

3. External Stylesheets

```
<style> rect {
  fill: purple;
  stroke: orange;
  stroke-width: 6;}</style>
```

4. SVG Attributes

```
fill="cyan"
```

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <style>
    rect {
      fill: purple;
      stroke: orange;
      stroke-width: 6;
    }
    </style>
</head>
<body>
<svg width="100" height="100">
    <style>
        rect {
          fill: rgb(0, 128, 0);
          stroke: yellow;
          stroke-width: 4;
        }
    </style>
    <rect x..y..width..fill="cyan" style="fill: red; stroke: blue; stroke-width: 2;" />
  </svg>

</body>
</html>
```

Demo: youtu.be/vjjnEHftuAE

SVG 1.1 Specification https://www.w3.org/TR/SVG11/styling.html

Styling Properties

SVG Styling Attributes	CSS Equivalent	JS Equivalent
height=”10rem”	height: 10rem;	svg.style.height = "10rem";
● height / width	● *	● *
● x / y	● *	● *
● fill	● fill	● fill
● cursor	● cursor	● cursor
● font-family	● fontFamily	● fontFamily
● font-size	● fontSize	● fontSize
● opacity	● opacity	● opacity
● stroke	● stroke	● stroke
● stroke-width	● strokeWidth	● strokeWidth
● transform	● transform	● transform

* No SVG 1.1 equivalent - use transform instead.

This works on JS for all Attributes:
svg.setAttribute('height', '10rem');

https://www.w3.org/TR/SVG/propidx.html

Sizing SVGs

- **width= or height= inside <svg>**
  - Overrides default.
- **width: or height: in CSS for <svg>**
  - Overrides <svg> attributes.
- **style= inside <svg>**
  - Overrides default.
  - Overrides other sizing.

- **viewBox only**
  - Defines aspect ratio of <svg>.
  - Defines scaling of <svg>.
- **Why the need for width= or height=**
  - Two kinds of browser behavior
    - **HTML**: "Default object size": width=300px, height=150px.
    - **SVG**: attribute default =100%.

Most reliable way, use viewBox.

helpful to add width and height, in addition to viewBox.

Demo: https://youtu.be/PLLgohjNUoQRm4MfqMwJnIaKcJyNaivpR5

Sizing SVGs

Browser	Mobile	Desktop
Chrome	HTML: default. SVG: default.	HTML: default. SVG: default.
Firefox	HTML: default. SVG: default.	HTML: default. SVG: default.
Edge	HTML: default. SVG: default.	HTML: default. SVG: default.

- Two kinds of browser **defaults**
  - **HTML**: “Default object size”: width=300px, height=150px.
  - **SVG**: attribute default =100%.

Animating SVGs

Use cases:

- Transitions
- Animations
- Interactivity

Techniques:

- SVG animate
- CSS Animations (@keyframes and animation)
- Javascript
- [SMIL (synchronised multimedia Integration language)]

```
<svg width="300" height="200">
    <rect id="myRect" x="50" y="50" width="100" height="100" fill="blue" onclick="changeColor()" />
    </rect>
        <animateTransform attributeName="transform"
                          attributeType="XML"
                          type="rotate"
                          dur="6s"
                          from="0 50 90"
                          to="360 0 0"
                          repeatCount="indefinite" />
    </rect>
  </svg>
```

```
#myRect {
    animation: scale 2s ease infinite alternate;
  }

  @keyframes scale {
    0% {
      transform: scale(1);
    }
    50% {
      transform: scale(1.5);
    }
    100% {
      transform: scale(1);
    }
  }

```

```
function changeColor() {
    var rect = document.getElementById('myRect');
    var currentColor = rect.getAttribute('fill');
    var newColor;
    switch (currentColor) {case 'blue': newColor = 'red';
        break;
      case 'red': newColor = 'green';
        break;
      case 'green': newColor = 'orange';
        break;
      case 'orange': newColor = 'blue';
        break;
      default: newColor = 'blue';
    }
  rect.setAttribute('fill', newColor);
}
```

Demo: youtu.be/EN23YNdoSlc?si=4-NJb-ZaVh2CDz3J

Use Cases

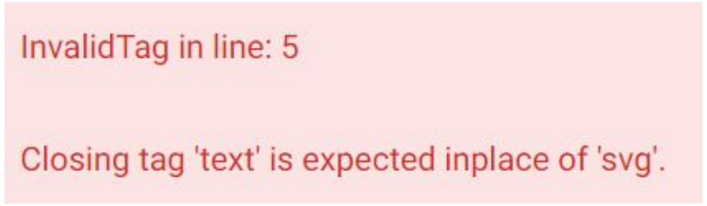
https://gitlab.tugraz.at/95FD77DBCF078A32/svg-on-the-web-use-cases

- Icons, with styling from outside.
- SVG Sprites
  - Multiple Icons in single SGV file (https://www.telerik.com/blogs/how-to-use-svg-react)
- Masking content
  - SVG as masking image with dynamic styling.
- Data driven graphs
  - High interactivity possible.
- Background image
- Situation-based icons
  - Clock / Weather / Moon phase

Validation of SVG 1.1

XMLLint - xmllint.com

- General XML Validator.
- Helpful for finding small mistakes.



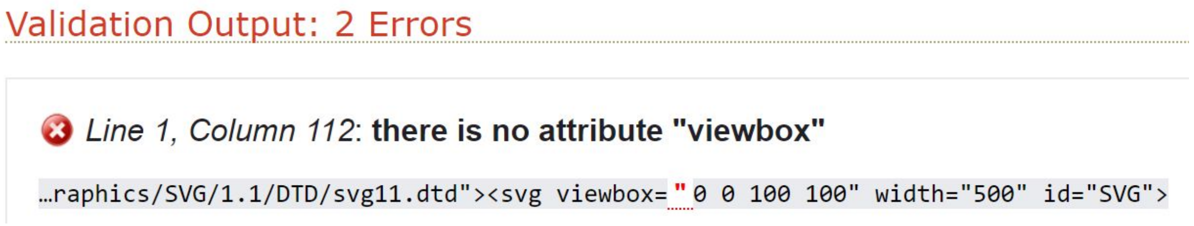
```
InvalidTag in line: 5

Closing tag 'text' is expected inplace of 'svg'.
```

XMLLint - Example error output.

W3C - validator.w3.org

- case-sensitive
- Very in-depth tips for fixing errors.



Validation Output: 2 Errors

Line 1, Column 112: **there is no attribute "viewbox"**

```
…raphics/SVG/1.1/DTD/svg11.dtd"><svg viewbox="0 0 100 100" width="500" id="SVG">
```