CSS and SVG

Understanding the structure of the SVG

The SVG produced by Verovio can be manipulated further. In this tutorial, you are going to use CSS to highlight some content of the output.

One key feature of Verovio is that it preserves the structure of the MEI in the SVG output. For example, a chord with two notes encoded in MEI:

```xml
<chord xml:id="c1">
  <note/>
  <note/>
</chord>
```

will have a the following structure in the SVG:

```xml
<g class="chord" id="c1">
  <g class="note"/>
  <g class="note"/>
</g>
```

You will notice that both the tree structure is preserved and that the MEI element names are passed as `@class` attribute values in the SVG elements, as well as the `@xml:id` of the MEI element as `@id` in the SVG.

Since SVG can be styled with CSS, it is straightforward to modify the appearance of elements and their contents.

Modifying the appearance can be done with a CSS file, or programmatically.

Applying CSS to the SVG

In the CSS file you need to create rules to be applied to the SVG `<g>` elements - simply `g` in CSS - together with the class selector corresponding to the MEI element name. For example, `g.tempo` modifies MEI `<tempo>` elements.

```
g.tempo {
  // ... some CSS properties
}
```

To modify the color, you need to change the `fill` property, and - in some cases - also the `color` property. The CSS rule will look like:

```
{
  fill: crimson;
  color: crimson;
}
```

You can also select elements based on their hierarchy. For example, you can select `<artic>` within `<chord>` with:

```
g.chord g.artic {
}
```

CSS can be animated, for example by making the colors pulsing. You need to specify an animation name with a duration and an iteration count together with corresponding key frames:

```
{
  animation-name: pulse;
  animation-duration: 1.0s;
  animation-iteration-count: infinite;
}

@keyframes pulse {
  0%   { fill: orange; }
  50%  { fill: brown; }
  100% { fill: orange; }
}
```

CSS can also be used to change the opacity of an element. The opacity value can range from 0.0 (transparent) to 1.0 (normal default value).

Adjusting the style programmatically

In applications, it is often useful to modify the CSS programmatically, for example in response to some user interactions.

To do so, elements can be accessed by element and class name in the same way as with CSS. For example, for retrieving all rests, you can do:

JAVASCRIPT

```
let rests = document.querySelectorAll('g.rest');
```

You can then loop through the list of rests with:

JAVASCRIPT

```
for (let rest of rests) {
    // you have now access to the rest one by one and can modify their style
}
```

To modify the style of an element, you can assign the desired value to the corresponding key. For example, in order to change the color (fill) or a rest (element), you need to do:

JAVASCRIPT

```
rest.style.fill = "dodgerblue";
```

Using custom data-* attributes

The attributes in the SVG `<g>` elements corresponding to the MEI elements is not limited to the `@class` carrying the MEI element name and the MEI `@xml:id` passed as `@id` . Verovio also passes MEI `@type` values as additional `@class` in the SVG.

However, in many cases, applications need to have access at other attribute values. To do so, one can use the `svgAdditionalAttribute` option to specify which attributes can be made available in the SVG output. For example, for making the note pitch name and the note octave accessible, you can add the following option values to Verovio's `setOptions()` method:

JSON

```
{
  svgAdditionalAttribute: ["note@pname", "note@oct"]
}
```

With this option, each `g.note` element in the SVG will also have a `data-pname` and a `data-oct` attribute carrying the original MEI attribute value. For example, a note in MEI and the corresponding SVG element will be:

XML

```
<note pname="c" oct="5"/>
```

XML

```
<g class="note" data-pname="c" data-oct="5"/>
```

This can be used in the query selector to restrict the matches to elements having specific attribute values. For example, for selecting the C5 notes, you would do:

JAVASCRIPT

```
let c5s = document.querySelectorAll('g[data-pname="c"][data-oct="5"]');
```

Accessing MEI attribute values programmatically

Custom `data-*` attributes are straightforward and easy to use with CSS selectors. However, selectors can have some limits, and it is not always possible to know in advance all the attributes that needed in the SVG. Furthermore, if the list of attributes becomes too long, the SVG might become overloaded.

In this case, it is possible and preferable to access them programmatically with JavaScript. This can be done through the `getElementAttr()` toolkit method that gives access to all the MEI attributes of a given element, including attributes not currently supported or not used by Verovio. It takes an `xml:id` value as the input parameter and returns a JSON object with all the attributes for that element from the MEI encoding. For example, given this MEI:

XML

```
<rest xml:id="r123" dur="4" dots="1">

```js
let attr = tk.getElementAttr("r123");
```

You can then look at any attributes specifically in the JSON object returned, for example `attr.dur` for the MEI `@dur` of the rest:

JAVASCRIPT

```
if (attr.dur && attr.dur == "1") {
  // This is a whole note rest
}
```

Full example

Open this example in a new window.

HTML / JAVASCRIPT
```

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width">
        <title>Understanding the structure of the SVG</title>
        <!-- A stylesheet for modifying the appearance of the output -->
        <link href="solution.css" rel="stylesheet" type="text/css" />
        <!-- Verovio -->
        <script src="https://www.verovio.org/javascript/develop/verovio-toolkit-wasm.js" defer></script>
    </head>
    <body>
        <script>
            document.addEventListener("DOMContentLoaded", (event) => {
                verovio.module.onRuntimeInitialized = function () {
                    // This line initializes the Verovio toolkit
                    const tk = new verovio.toolkit();

                    let zoom = 80;
                    let pageHeight = document.body.clientHeight * 100 / zoom;
                    let pageWidth = document.body.clientWidth * 100 / zoom;

                    options = {
                        pageHeight: pageHeight,
                        pageWidth: pageWidth,
                        scale: zoom,
                        // Add an option to pass note@pname and note@oct as svg @data-*
                        svgAdditionalAttribute: ["note@pname", "note@oct"]
                    };
                    tk.setOptions(options);

                    // This line fetches the MEI file we want to render...
                    fetch('https://www.verovio.org/examples/downloads/Schubert_Lindenbaum.mei')
                        // ... then receives the response and "unpacks" the MEI from it
                        .then((response) => response.text())
                        .then((mei) => {
                            // ... then we can load the data into Verovio ...
                            tk.loadData(mei);
                            // ... and generate the SVG for the first page
                            let svg = tk.renderToSVG(1);
                            // ... and finally gets the <div> element with the ID we specified,
                            // and sets the content (innerHTML) to the SVG that we just generated.
                            document.getElementById("notation").innerHTML = svg;

                            // Get all the rests by selecting <g> with attribute class 'rest' ...
                            let rests = document.querySelectorAll('g.rest');
                            // ... and change their color by setting their style.fill value
                            for (let rest of rests) {
                                rest.style.fill = "dodgerblue";
                            }

                            // Get all the notes with @pname="c" and @oct="5" and change their color
                            let c5s = document.querySelectorAll('g[data-pname="c"][data-oct="5"]');
                            for (let c5 of c5s) {
                                c5.style.fill = "aqua";
                            }

                            // Get all the verses ...
                            let verses = document.querySelectorAll('g.verse');
                            // ... and use the 'getElementAttr()' to retrieve all attributes ...
                            for (let verse of verses) {
```

```
                                let attr = tk.getElementAttr(verse.id);
                                // ... and change to color when @n exists and is greater than 1
                                if (attr.n && attr.n > 1) verse.style.fill = "darkcyan";
                            }
                        });
                }
            });
        </script>
        <!-- The div where we are going to insert the SVG -->
        <div id="notation" />
    </body>
</html>
```