Add interactivity to the cap

1/2 Pseudo code

Fetch the SVG put it in a html element.

Make a global variable call it `elementToPaint`
Give a class (for example `g_to_interact_with`) to the g elements you want to interact with (Visor and Crown) in the SVG. If you at some point in another assignment should have many g elements, you can use Visual Studios search and replace capabilities.

SVG: remove fill information from the paths, otherwise we can’t fill them by putting fill on the `g` - element

Add `pointer-events: none;` CSS rule to the shadows layer, otherwise we can’t mouse through it.

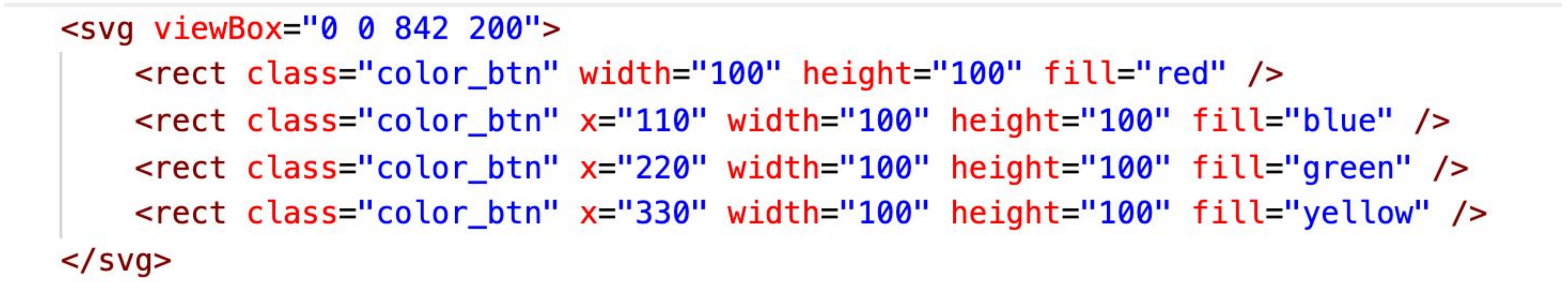
Add mouse events to relevant g elements (`g_to_interact_with`) with querySelectorAll foreach

On click save the current element (`this`) in to your global variable and set the fill of the current element to grey so we know that it’s chosen for coloring.

2/2 Pseudo code

On mouseover and mouseout highlight/unhighlight the element, could be the path for example.

Add this (non pseudo 🙂) code beneath the element holding the fetched SVG



```
<svg viewBox="0 0 842 200">
    <rect class="color_btn" width="100" height="100" fill="red" />
    <rect class="color_btn" x="110" width="100" height="100" fill="blue" />
    <rect class="color_btn" x="220" width="100" height="100" fill="green" />
    <rect class="color_btn" x="330" width="100" height="100" fill="yellow" />
</svg>
```

Add eventlisteners to these buttons. Make them add the clicked buttons fill color to the element stored in the `elementToPaint` variable, `if` it's not `undefined`!