Objectives

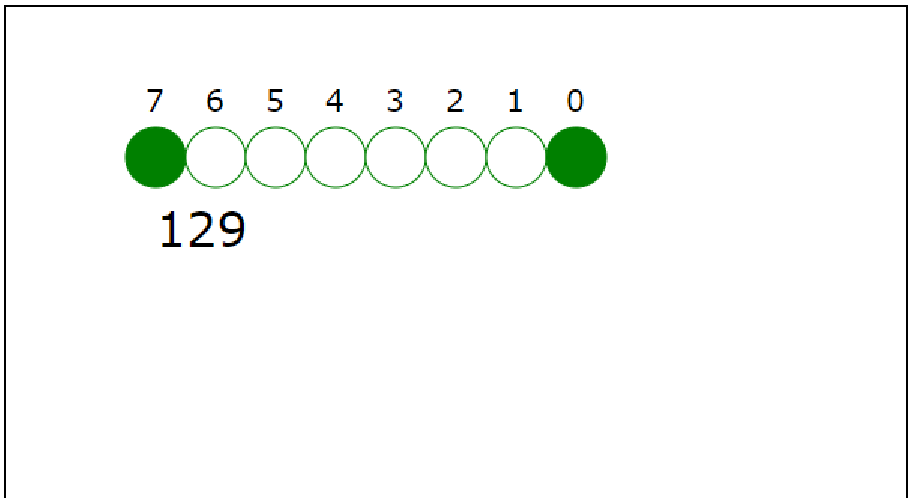
- Practice creating and modifying SVG elements using the JavaScript browser API.
- Listen for and react to events dispatched by SVG elements.

Instructions

Write an interactive graphical byte-to-decimal converter in JavaScript. Visualize the bits of a byte as eight SVG shape elements in a row, where each shape represents the value of one bit. When a shape is “on” the corresponding bit has the value 1, and when it is “off” its value is 0. The value of a bit should be represented graphically, such as by setting a shape’s fill color. Clicking on a shape should toggle its value between “on” and “off”, (i.e. 1 and 0).

The value of each shape contributes a power of 2 to the numerical value of the byte. The right-most shape is the least significant bit which contributes 2^0=1 to the byte. The second-from-right contributes 2^1=2, and so one through the most significant left-most bit which contributes 2^7=128. Sum all contributed powers of 2 to determine the final byte value.

Below the row of shapes place an SVG text element to show the current decimal value of the byte currently displayed graphically. Each time a shape is clicked its value is toggled and a new byte value is computed and displayed by the text element. Following is an example. I added text elements above each shape to indicate the power of 2 contributed by the corresponding bit – this is not required.



1. You may start with the Lab06_start.html file, which is an HTML file containing a base <svg> element with the id “cvs” and an empty <script> element for your JavaScript code.
2. Your solution must define at least three functions named init(), toggle(event), and update().
    - The init() function creates and appends all SVG elements to <svg>. The click event dispatched by each SVG element representing a bit should be configured to invoke the toggle() function.
    - The toggle(event) function should toggle the stored bit value represented by the event’s target shape and invoke update() to modify the graphical representation of the byte.
    - The update() method should modify the graphical representation to match the stored bits, compute a new byte value based on those bits, and update the displayed byte value.
3. Wrap ALL your code in an IIFE. Your code must not define any new properties in the global object.

4. You may use the following helper function to create new SVG elements.

```
// Helper function for creating SVG elements.
// Sets namespace URI so we don't have to think about it.
// @param  {string}  tag       : the name of the SVG element tag
// @param  {object}  attrs     : object properties/values set as SVG attributes
// @param  {string}  innerHTML : if provided, set as the new element's inner HTML
// @return {element} newly created SVG element
function createSVGElement(tag, attrs, innerHTML) {
  let ns = "http://www.w3.org/2000/svg";
  let el = document.createElementNS(ns, tag);
  for (let attr in attrs) { el.setAttribute(attr, String(attrs[attr])); }
  if (innerHTML) { el.innerHTML = innerHTML; }
  return el;
}
```

Hints

- When reading or writing attributes of an SVG element, do not use the property assignment shortcut commonly used for HTML elements. Instead, use the getAttribute() and setAttribute() Element methods to get/set attributes.
- Consider using an internal Array of eight numbers (0's or 1's) to represent bits of your byte. It is not necessary to use bitwise operators.
- Refer to the SVG element reference and the SVG attribute reference in the MDN SVG documentation for details.

Finishing Up

- You MUST enter header comments into your JavaScript file including (1) File name, (2) Your name, (3) Description and or purpose of the assignment
- You MUST comment your code, explaining what your code does in each section.
- Submit your file(s) using Canvas under the appropriate assignment name.