Style Interaction

I've been working on a project where I want to convey the idea of pages spread out on a table or post-it notes pasted randomly against a wall or another surface. It took me a while to reason through it but I ended happy with the result. It also shows the interaction between stylesheets and inline styles generated with JavaScript.

The HTML for the project is fairly simple. It doesn't matter what's inside the story elements... for the purpose of the movement we only care about the story elements.

```
<div class="story-container">
  <div class="story">
  </div>
<div class="story">y">
  </div>

  <div class="story">
  </div>

  <div class="story">
  </div>
</div> </div><!-- closes story-container -->
```

The first block of CSS provides styles for the container, the initial state for stories (the `.story` class declaration), and state for the static elements (the `.absolute` class declaration)

Some things two notice.

The container has been absolutely positioned (`position: absolute`) and the stories have a relative position (`position: relative`). We did this to make sure that the stories can be relatively positioned and therefore have their location in the page manipulated.

The absolutely positioned elements will be placed at the top of the screen, covering any other elements.

```
.story-container {
  position: absolute;
  width: 90vw;
  margin: 0 auto;
}

/* Initial state */
.story {
  position: relative;
  background-color: lightgoldenrodyellow;
  color: black;
  border: 2px solid black;
  border-radius: 15px;
  width: 50%;
  padding: 1em;
  margin: 1em 0;
}

/* state when clicked */
.absolute {
  position: absolute;
  background-color: lightgoldenrodyellow;
  color: black;
  border: 2px solid black;
  border-radius: 15px;
  width: 80%;
  padding: 1em;
  margin: 1em 0;
  z-index: 100;
}
```

The second block is where we position our stories. The SCSS files use functions to generate random values for each of the three elements we transform, rotation in the Z axis (`rotateZ`), movement in the X axis (`translateX`), and movement in the Y axis (`translateY`).

Because of limitations on how SASS works with random numbers, calculated at compile time rather than runtime, we run the risk of getting the same numbers applied to each instance. That's why we create different rules for different stories.

The full example has 7 different rules that place the corresponding elements in random locations.

```
/* Position of the stories */
.story:first-child {
  transform-origin: top;
  transform: rotateZ(-29deg) translateX(52px) translateY(50px);
  z-index: 26;
}

.story:nth-child(2n) {
  transform-origin: top;
  transform: rotateZ(-29deg) translateX(55px) translateY(25px);
  z-index: 24;
}
```

JavaScript is where the magic happens.

We create an array with all the elements with class story (`.story`) using array.from.

Once we have the array we walk through it using a for loop. In the loop we use a click event handler to toggle both the story and absolute classes using classlist.toggle.

Toggle will add a class when it's not present and remove it if it exists for the element. Since all the elements initially are story, it will be removed for the element we clicked on. No element has the absolute class to begin with so it will be added at the same time the story class is removed.

```
const stories = Array.from(document.querySelectorAll('.story'));

for (let i = 0; i < stories.length; i++) {
  stories[i].addEventListener("click", function () {
    stories[i].classList.toggle("story");
    stories[i].classList.toggle("absolute");
  });
}
```

The last bit to remember. Javascript doesn’t write to an existing stylesheet but adds the styles as an inline style attribute. This is important if you have other CSS rules that apply to the element and may cause cascade issues.

See Cascade and inheritance at MDN for a more thoroguh discussion of the cascade and inheritance in CSS.