<erom> - 각 요소별 순차 스크롤 이벤트


```
//스크롤이벤트_뉴스
    window.addEventListener("scroll", function() {
        let newsBox1 = document.querySelector(".newsBox0401")
        let news01 = newsBox1.querySelector(".news01")
        let news02 = newsBox1.querySelector(".news02")
        let news03 = newsBox1.querySelector(".news03")
        let svg = newsBox1.querySelector(".newsBox0401 > div:nth-child(5)")

        let newsBox2 = document.querySelector(".newsBox0402")
        let lines = newsBox2.querySelectorAll(".line0401")
        let textBoxes = newsBox2.querySelectorAll("[class^='textBox']")

        let trigger = window.innerHeight * 0.8
        let scrollY = window.scrollY

        if (scrollY + trigger > newsBox1.offsetTop) {
            // 뉴스 박스1 순차 애니메이션
            [news01, news02, news03, svg].forEach((el, i) => {
                el.style.transitionDelay = `${i * 0.3}s`
                el.classList.add(i < 2 ? "activeNews" : "activeNews2")
            })

            // 뉴스 박스2 라인 + 텍스트 순차 애니메이션
            lines.forEach((line, i) => {
                line.style.transitionDelay = `${i * 0.4}s`
                line.classList.add("activeLine")
            })
            textBoxes.forEach((box, i) => {
                box.style.transitionDelay = `${(i + 1) * 0.6}s`
```

```
                box.classList.add("active3")
            })
        } else {

            [news01, news02, news03, svg].forEach(el => {
                el.classList.remove("activeNews", "activeNews2")
                el.style.transitionDelay = "0s"
            })
            lines.forEach(line => {
                line.classList.remove("activeLine")
                line.style.transitionDelay = "0s"
            })
            textBoxes.forEach(box => {
                box.classList.remove("active3")
                box.style.transitionDelay = "0s"
            })
        }
    })
```

한 섹션안에 회전효과가 필요한 요소와 슬라이딩효과가 필요한 요소가 섞여있을 뿐더러 요소가 많아서 자연스러운 흐름을 만들기가 힘들었다.

최대한 자연스러운 스크롤 효과를 내기 위해서 실제 브라우저화면(뷰포트)의 높이를 나타내는 inerHeight를 사용했다. 해당 섹션의 스크롤 이벤트가 시간적으로 길고, 실제 여러 비율을 대입해보니 시각적으로 섹션이 눈에 들어오기 시작하는 *0.8의 비율을 적용했다.

우선 화면 중앙을 기준으로 좌측 요소들은 처음에는 회전효과가 나타나는 요소와 슬라이딩효과가 나타나는 요소를 따로따로 처리하였었으나, 코드의 편의성을 고려하고 (같은 박스안의 요소인데 따로 해놓으니 헷갈렸음.) 삼항연산자를 사용함으로써 한 번에 ForEach로 처리하는 것이 가능했다.
index와 삼항연산자를 활용하여 회전효과가 적용되는 첫 번째 요소와 두번째 요소, 슬라이딩 효과가 적용되는 세번째와 네 번째 요소가 한 번에 처리되도록 했다.

우측 요소들은 라인과 텍스트가 섞여있어서 무작정 순차적으로 나타나는 것보다는 라인과 텍스트를 구분하여 심미적으로 아름답고 자연스러운 애니메이션을 만드는데 집중했다.
요소를 불러올 때 선택자를 활용하여 라인과 텍스트를 각각의 배열로 가져와 딜레이를 따로 주어 자연스러운 흐름을 만들었다.