Dragula

드래그 이벤트 사용 시 유용한 라이브러리 중 하나

```
<script>
  dragula([document.getElementById('container')], {
    moves: function (el, container, handle) {
      return handle.classList.contains('move');
    }
  });
</script>
```

스크립트 코드에서는 dragula 의 selector와 option 을 지정 합니다.

이때, selector는 drag 시키려는 요소의 바로 부모여야 적용 됩니다.(현재 테스트까진...)

option은 안줘도 되는데, 여기서는 `li` 안에 `span` 을 주고 해당 `span`부분만 눌러 이동 이벤트를 주었습니다.

```
.gu-mirror{display: none;}
```

dragula 에서는 `gu-mirror` 라는 태그가 만들어집니다.

바로 보이진 않고 요소 선택 후 드래그 시 화면 하단에 노출됩니다.

이를 노출시키지 않기 위해 `display:none` 처리 했습니다

```
<div class="dragula">
  <ul id="container">
    <li>Clicking on these elements triggers a regular <code>click</code> event you can listen to.<span class="move">*</span></li>
    <li>Try dragging or clicking on this element.<span class="move">*</span></li>
    <li>Note how you can click normally?<span class="move">*</span></li>
    <li>Drags don't trigger click events.<span class="move">*</span></li>
    <li>Clicks don't end up in a drag, either.<span class="move">*</span></li>
    <li>This is useful if you have elements that can be both clicked or dragged.<span class="move">*</span></li>
  </ul>
</div>
```

마크업 구조를 보면 div로 감싸준 후 `ul` 에 `id` `container` 를 주었습니다. id 는 다른 이름으로 줘도 무방합니다.

그리고 이동 요소가 되야하는 `li` 를 만들었습니다.

스크립트에서 move 클래스들에 이벤트를 주도록 했기 때문에 `li` 안에 `<span class="move">` 를 적용했습니다.(안주면 li 전체에 이동 이벤트가 적용됩니다)

이대로 하면 drag 이벤트가 작동합니다.

물론 라이브러리는 선언해야합니다.(cdn/npm)