6주차

생성일	@2023년 8월 10일 오후 9:47

throttle, debounce

scroll, input, mouse move 같은 이벤트는 짧은 시간동안 여러번 발생해 성능 문제가 생길 수 있다.

이벤트를 제한해 성능 문제를 해결할 수 있다.

- throttle
  - 이벤트를 일정 주기마다 발생하게 함

```
function throttle(callback, time) {
  let timer;

  return function () {
    if (!timer) {
      timer = setTimeout(() => {
        callback.apply(this, arguments);
        timer = undefined;
      }, time);
    }
  };
}
```

- debounce
  - 이벤트를 그룹화해 제일 처음 또는 제일 마지막 이벤트만 실행함

```
function debounce(callback, delay) {
  let timer;

  return function () {
    if (timer) {
      clearTimeout(timer);
    }
    timer = setTimeout(() => callback.apply(this, arguments), delay);
  };
}
```

- test html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Document</title>
  </head>
  <body>
    <input id="input" />
    <h3>none</h3>
    <div id="none"></div>
    <br />
    <h3>throttling</h3>
    <div id="throttle"></div>
    <br />
    <h3>debouncing</h3>
    <div id="debounce"></div>
    <script>
      const inputEl = document.querySelector("#input");
      const noneEl = document.querySelector("#none");
      const throttleEl = document.querySelector("#throttle");
      const debounceEl = document.querySelector("#debounce");

      function throttle(callback, time) {
        let timer;
```

```
        return function () {
          if (!timer) {
            timer = setTimeout(() => {
              callback.apply(this, arguments);
              timer = undefined;
            }, time);
          }
        };
      }

      function debounce(callback, delay) {
        let timer;

        return function () {
          if (timer) {
            clearTimeout(timer);
          }
          timer = setTimeout(() => callback.apply(this, arguments), delay);
        };
      }

      function normalInputHandler(e) {
        none.innerText = e.target.value;
      }

      function throttleInputHandler(e) {
        throttleEl.innerText = e.target.value;
      }

      function debounceInputHandler(e) {
        debounceEl.innerText = e.target.value;
      }

      inputEl.addEventListener("input", normalInputHandler);
      inputEl.addEventListener("input", throttle(throttleInputHandler, 500));
      inputEl.addEventListener("input", debounce(debounceInputHandler, 500));
    </script>
  </body>
</html>
```

javascript의 문법적 설탕

문법적 설탕 : 코드의 가독성을 높이거나 간결하게 표현하기 위한 문법

- nullish coalescing operator(null 병합 연산자)

```
//before
const something = null;
let text;

if (!something) {
  text = "default value1";
}
console.log(text);  //default value1

//after
text = something ?? "defualt value2";
console.log(text);  //default value2
```

- spread syntax

```
const arr1 = [1,2,3];
const arr2 = [4,5,6];

//before
let newArr = arr1.concat(arr2);   //[ 1, 2, 3, 4, 5, 6 ]

//after
newArr = [...arr1, ...arr2];    [ 1, 2, 3, 4, 5, 6 ]
```