6ая ЛР продолжает 5ую работу.

В 6ой работе необходимо обрабатывать события сенсорного экрана (*touchstart, touchend*).

Касание должно работать как и клик мышью (т.е. должна быть возможность перетаскивать элемент)

После двойного нажатия на сенсорный экран *div* переходит в режим «следующий за пальцем». Когда вы делаете *touchstart*, *div* следует за пальцем, даже если это прикосновение не к объекту, до *touchend*. Однако *touchend* не останавливает режим, поэтому следующий *touchstart* продолжает перемещать тот же *div* (опять же, даже если *touchstart* не находится на *div*). Этот режим заканчивается только действием щелчка (*touchstart* и *touchend* быстро в одном и том же месте).

Если во время перетаскивания на экран опускается второй палец, это событие должно обрабатываться по аналогии с клавишей *esc* (т.е. перетаскивание прекращается, элемент возвращается на то место откуда началось перемещение).

Доп задание: Реализовать изменение размера объекта, предусмотреть минимально возможный размер элемента.

const targets = document.querySelectorAll('.target');

let isDragging = false;

let isPinned = false;

let currentElement = null;

let offsetX, offsetY;

let originalPosition = {};

let initialTouchDistance = 0;

let minSize = 50; // Минимальный размер элемента

// Функция для начала перетаскивания

function startDrag(event) {

if (!isPinned) {

isDragging = true;

currentElement = event.target;

offsetX = event.clientX - currentElement.getBoundingClientRect().left;

offsetY = event.clientY - currentElement.getBoundingClientRect().top;

}

}

// Функция для выполнения перетаскивания

function drag(event) {

if (isDragging && currentElement) {

currentElement.style.position = 'absolute';

currentElement.style.left = `${event.clientX - offsetX}px`;

currentElement.style.top = `${event.clientY - offsetY}px`;

}

}

// Функция для завершения перетаскивания

function endDrag() {

isDragging = false;

currentElement = null;

}

// Функция для двойного клика

function doubleClick(event) {

isPinned = true;

currentElement = event.target;

originalPosition = {

left: currentElement.style.left || '0px',

top: currentElement.style.top || '0px',

};

currentElement.style.backgroundColor = 'lightblue'; // меняем цвет

}

// Функция для отключения пиннинга

function unpin() {

if (currentElement) {

isPinned = false;

currentElement.style.backgroundColor = ''; // сбрасываем цвет

currentElement = null;

}

}

// Функция для обработки нажатия клавиши Esc

function handleKeyDown(event) {

if (event.key === 'Escape' && currentElement) {

currentElement.style.left = originalPosition.left;

currentElement.style.top = originalPosition.top;

unpin();

}

}

// Обработка сенсорных событий

function touchStart(event) {

event.preventDefault();

const touch = event.touches[0];

if (!isPinned) {

isDragging = true;

currentElement = event.target;

offsetX = touch.clientX - currentElement.getBoundingClientRect().left;

offsetY = touch.clientY - currentElement.getBoundingClientRect().top;

} else {

// Пиннинг: начинаем следовать за пальцем

dragTouch(touch);

}

}

function dragTouch(touch) {

if (currentElement) {

currentElement.style.position = 'absolute';

currentElement.style.left = `${touch.clientX - offsetX}px`;

currentElement.style.top = `${touch.clientY - offsetY}px`;

}

}

function touchEnd(event) {

if (isDragging) {

isDragging = false;

currentElement = null;

}

}

// Обработка изменения размера

function handlePinch(event) {

if (event.touches.length === 2) {

const dx = event.touches[1].clientX - event.touches[0].clientX;

const dy = event.touches[1].clientY - event.touches[0].clientY;

const distance = Math.sqrt(dx \* dx + dy \* dy);

if (initialTouchDistance === 0) {

initialTouchDistance = distance;

} else {

const scale = distance / initialTouchDistance;

const newWidth = Math.max(minSize, currentElement.offsetWidth \* scale);

const newHeight = Math.max(minSize, currentElement.offsetHeight \* scale);

currentElement.style.width = `${newWidth}px`;

currentElement.style.height = `${newHeight}px`;

}

}

}

function handleTouchCancel(event) {

initialTouchDistance = 0; // Сброс расстояния при отмене

}

// Добавляем обработчики событий к каждому элементу

targets.forEach(target => {

target.addEventListener('mousedown', startDrag);

target.addEventListener('dblclick', doubleClick);

target.addEventListener('touchstart', touchStart);

target.addEventListener('touchend', touchEnd);

target.addEventListener('touchmove', (event) => {

if (isPinned) {

dragTouch(event.touches[0]);

} else {

drag(event);

}

});

});

document.addEventListener('mousemove', drag);

document.addEventListener('mouseup', endDrag);

document.addEventListener('click', unpin);

document.addEventListener('keydown', handleKeyDown);

document.addEventListener('touchmove', handlePinch);

document.addEventListener('touchcancel', handleTouchCancel);