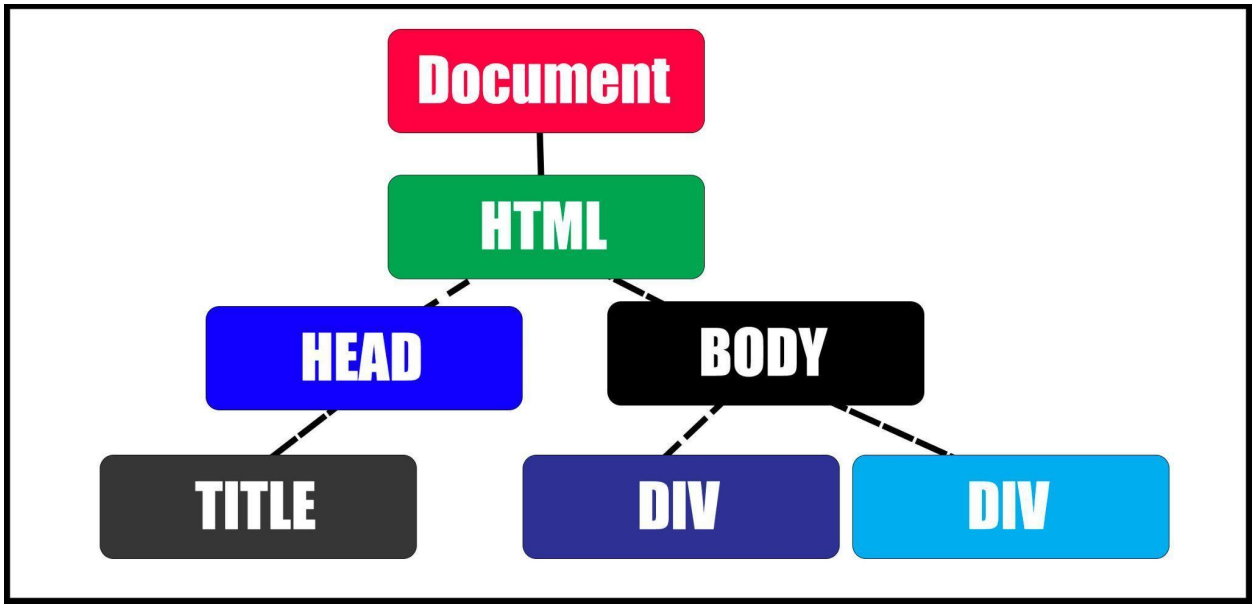
Dynamic Interactive Web Pages with JavaScript

Introduction to JavaScript and the DOM how to create interactive web pages

The Document Object Model (DOM) is a programming interface that can be accessed with JavaScript to connect to web document contents. Web documents are based on HTML structure and can be viewed in a browser that renders the HTML or as HTML code. The HTML code is used to construct the DOM, which will represent the HTML source that the browser will use to construct the page. The DOM is used to create the web page, and selecting and updating the DOM representation of the HTML code allows JavaScript to make changes to the HTML without changing the HTML source code. The DOM is a standard object that has the HTML elements as objects, each with properties, methods and events.

Using JavaScript to connect to the DOM Object by selecting Page elements allows you to interact with them. Explore how to access page elements, manipulate content and properties of the page element. With JavaScript select and update contents of the element. DOM is a tree structure representing all the page elements and properties of those elements.



Using JavaScript you can change HTML elements, update attributes, remove and add HTML elements, add events.

Introduction to console.dir() document.querySelector() document.querySelectorAll() document.getElementtextContent() property value assignment.

document.querySelectorAll() returns a NodeList
document.getElementsByTagName() and document.getElementsByClassName() returns an HTMLCollection

Nodes in the DOM are referred to as all items, so everything is a node and every node is an object.

NodeList

- Items can be selected by the index value that look like an array but are not
- NodeList items can only be accessed with their index value
- returns a static NodeList
- You can access the childNodes of the list which is live
- list with a length property which can be looped through
- can contain any node type

HTMLCollection

- Returns elements with ID in the collection, which can be selected using the ID from the list as well as index value.
- HTMLCollection items can be accessed with a name, index or id value.
- Live collection, does not include text nodes
- item method can be used to access the elements from the list. eles.item(0)

- list with a length property which can be looped through
- contain any node Element

```
                                                                    apps1.js:18
▶ NodeList(4) [div#one.one, div#two.one, div#three.two, div.one]
                                                                    apps1.js:20
▶ HTMLCollection(4) [div#one.one, div#two.one, div#three.two, div.one,
one: div#one.one, two: div#two.one, three: div#three.two]
                                                                    apps1.js:21
▶ HTMLCollection(3) [div#one.one, div#two.one, div.one, one:
div#one.one, two: div#two.one]
```

Lesson Coding Exercise :

1. Create an HTML file with page elements, include classes, and ids in the elements.
2. Select the elements using various methods .
3. Loop through a nodeList and an HTMLcollection output the elements and update the innerHTML of them. Observe the difference between the NodeList and HTMLCollection.

```
<!doctype html>
<html>
<head>
  <title>JavaScript</title>
</head>
<body>
 <div class="one" id="one">JavaScript 1</div>
 <div class="one" id="two">JavaScript 2</div>
 <div class="two" id="three"><div  class="one" >JavaScript
3</div></div>
 <script src="apps1.js"></script>
</body>
</html>
```

```
const ele1 = document.querySelector('.one');
const ele2 = document.querySelector('#two');
const ele3 = document.querySelector('div > div');
```

```
const ele4 = document.getElementById('one');
const eles1 = document.querySelectorAll('div');
const eles2 = document.getElementsByTagName('div');
const eles3 = document.getElementsByClassName('one');

console.log(ele1);
console.log(ele2);
console.log(ele3);
console.log(ele4);
console.log(eles2['one']);
console.clear();
console.log(eles1);

console.log(eles2);
console.log(eles3);

eles1.forEach((ele,ind,arr)=>{
   console.log(ele.textContent);
   ele.innerHTML = `<div>JavaScript List ${ind+1} </div>`;
   //console.log(arr);
})
console.log(eles1);
console.log(eles2);
for(let i=0;i<eles2.length;i++){
   console.log(eles2[i].innerHTML);
}

//console.log(eles2.item(1));
```

Creating and Deleting Elements with JavaScript Code

Lesson Coding Exercise :

1. Select HTMl elements to be used as parent elements for elements that will be created using document.createElement()
2. Create a function that can be used to generate new page elements. In the parameters add the element tag type, the parent and the html or contents of the element.
3. Using the function create an input element and a button element
4. Set the attribute of the input to be text and value of a string
5. Add an onclick event to the button, when clicked it will add a new item to the list
6. Create a list appended to the page. Using a for loop create several list items
7. Create a function that will add a list item to the unordered list. Update the inner text with a value, and add a style color for the text.
8. Within the function to create the list items, add an onclick event to the list items, that will select the list item and remove it from the parent. removeChild()
9. When the button is clicked add a function that will add a new list item, count the current number of list items. Add to the text of the new list item, the current input field value and the count of current list items in the DOM. Set a red font color in the new list item.
10. Check out the difference between append() prepend() appendChild() methods

```
<!doctype html>
<html>
<head>
  <title>JavaScript</title>
</head>
<body>
 <div class="one" id="one">JavaScript 1</div>
 <div class="one" id="two">JavaScript 2</div>
 <div class="two" id="three"><div class="one" >JavaScript
3</div></div>
```

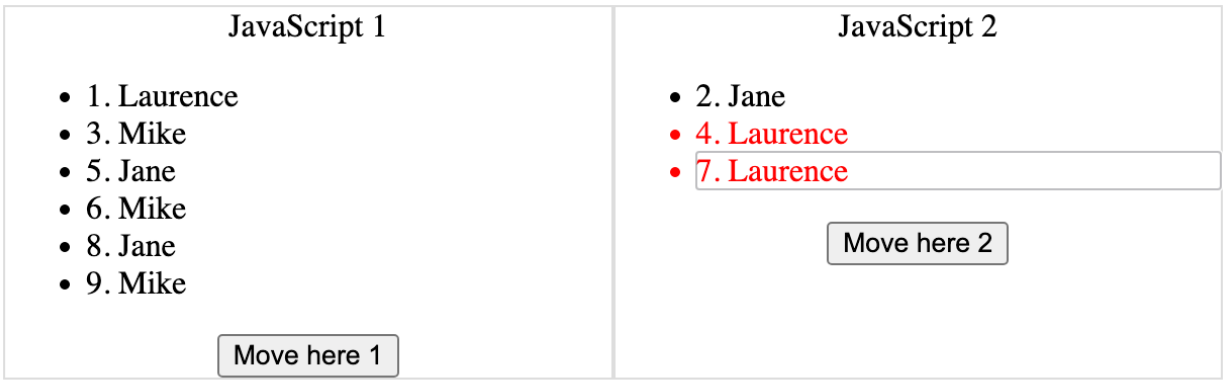
```
 <script src="apps2.js"></script>
</body>
</html>
```

```
const div1 = document.querySelector('div');
const div5 = document.querySelector('#three');
const div6 = document.querySelector('div .one');
const ul1 = document.createElement('ul');
const myInput = maker('input',div1,'');
myInput.setAttribute('type','text');
myInput.value = 'Laurence Svekis';
const btn = maker('button',div1,'Add to List');
btn.onclick = ()=>{
   const lis = document.querySelectorAll('li');
   let temp =  `${myInput.value} ${lis.length+1}`;
   addListItem(temp,'red');
}

div6.style.color = 'purple';
div6.textContent = 'Hello WOrld 2';

console.log(div6.parentNode);
div6.parentNode.removeChild(div6);

console.log(div1);



div1.append(ul1);
console.log(ul1);
```

```
for(let i=0;i<10;i++){
   addListItem(`List item ${i+1}`,'blue');
}

function addListItem(val,col){
   const li = document.createElement('li');
   li.innerText = val;
   li.style.color = col;
   li.onclick = (e)=>{
       li.parentNode.removeChild(li);
   }
   return ul1.appendChild(li);
}

const div2 = document.createElement('div');
div1.prepend(div2);
div2.innerHTML = `Hello World`;

const div3 = maker('div',div1,'Laurence Svekis');
div3.style.backgroundColor = 'red';
div5.append(div3);
div5.prepend(ul1);
div5.append(div6);

function maker(eleT,parent,html){
   const ele = document.createElement(eleT);
   ele.innerHTML = html;
```

```
    return parent.appendChild(ele);
}
```

Lists Move Elements from Parent to other element Items to Another List JavaScript Example



Lesson Coding Exercise :

1. update the parent styling to add CSS grid. document.body.style.display = 'grid' and set 2 column display of the children document.body.style.gridTemplateColumns = '1fr 1fr'
2. Create an array with string values
3. Create a global array which can be used to hold the ul lists
4. Select the divs from the page. Loop through each div and update the textContent to the list with index as a count.
5. Create an ul and add it into the div. Update the div with styling and append the ul to the parent div.
6. Add the new ul list to the uls global array
7. On the first run of the list append all the items from the array as new list items in the ul.
8. For the list items to make the HTML element editable add the attribute setAttribute('contenteditable', true)
9. Add a click event that toggles the class of the element with a class sel. Append the new list items to the parent ul
10. Add a button to the bottom of each list, add text content to the button.
11. When the button is clicked add an event to run a function called mover.

12. In the mover function, select the parent node of the button, using querySelect select the ul in the parent of the button.
13. Select all the elements that have a class of sel.
14. Using replaceChildren(...eles, ...parentEle.children) replace the children of the ul with the selected elements, add the existing elements from the parent back into the ul.
15. Remove the sel class from all the elements that have it.

```
<!doctype html>
<html>
<head>
  <title>JavaScript</title>
  <style>
     .sel{
        color:red;
     }
  </style>
</head>
<body>
 <div >JavaScript 1</div>
 <div >JavaScript 2</div>
 <div >JavaScript 3</div>
 <div >JavaScript 4</div>
 <script src="apps3.js"></script>
</body>
</html>
```

```
const divs = document.querySelectorAll('div');
document.body.style.display = 'grid';
document.body.style.gridTemplateColumns = '1fr 1fr';
const arr = ['Laurence', 'Jane', 'Mike', 'Laurence', 'Jane',
'Mike', 'Laurence', 'Jane', 'Mike'];
```

```
const uls = [];
console.log(divs);

divs.forEach((ele, ind) => {
   ele.textContent = `List #${ind+1}`;
   const ul = document.createElement('ul');
   ele.style.border = '1px solid #ddd';
   ele.style.textAlign = 'center';
   ele.append(ul);
   uls.push(ul);
   if (ind == 0) addItems();
   const btn = document.createElement('button');
   ele.append(btn);
   btn.textContent = `Move here ${ind+1}`;
   btn.onclick = mover;
})

function mover(e) {

   const parentEle = e.target.parentNode.querySelector('ul');
   console.log(parentEle.children);
   const eles = document.querySelectorAll('.sel');
   console.log(eles);
   parentEle.replaceChildren(...eles, ...parentEle.children);
   eles.forEach(ele => {
       ele.classList.remove('sel');
   })
}
```

```
function addItems() {
   arr.forEach((val, ind) => {
       const li = document.createElement('li');
       li.textContent = `${ind+1}. ${val}`;
       li.setAttribute('contenteditable', true);
       li.style.textAlign = 'left';
       li.onclick = () => {
           li.classList.toggle('sel');
           /*
                      console.log(li.classList.contains('sel'));

           if(li.classList.contains('sel')){
               li.classList.remove('sel');
           }else{
               li.classList.add('sel');
           }
           */

       }
       uls[0].append(li);
   })
}
```

AddEventListeners OnLoad and DOM loading events in JavaScript

Lesson Coding Exercise :

1. Add an event that gets invoked when the DOM content loads. Try window.onload , document.body.onload and document.addEventListener('DOMContentLoaded',()=>{})
2. Using addEventListener() add multiple events to the same element, try it also with onclick
3. Nest an element within its parent, using the useCapture set the boolean to true and check out the difference in order.
4. Add options to the event to only run once. {once: true}
5. Remove an event listener from an element removeEventListener

```
<!doctype html>
<html>
<head>
  <title>JavaScript</title>
  <script src="apps4.js"></script>
</head>
<body>
 <div >JavaScript 1</div>


</body>
</html>
```

```
window.onload = init;
//document.body.onload = init1;
document.addEventListener('DOMContentLoaded', init3);


function init3() {
   const div = document.querySelector('div');
   div.style.backgroundColor = '#ddd';
```

```
    const h1 = document.createElement('h1');
    h1.textContent = 'Laurence Svekis';
    div.append(h1);
    div.addEventListener('click', click1, {
        once: true
    });
    h1.addEventListener('click', click2);
    h1.addEventListener('click', click3);
}


function init2() {
    const div = document.querySelector('div');
    div.style.backgroundColor = '#ddd';
    const h1 = document.createElement('h1');
    h1.textContent = 'Laurence Svekis';
    div.append(h1);
    div.onclick = click1;
    h1.onclick = click2;
}


function click1(e) {
    //console.log(e.target);
    console.log('DIV');
}

function click2(e) {
```

```
    //console.log(e.target);
    //e.target.removeEventListener('click',click2);
    console.log('H1');
}

function click3(e) {
    e.target.removeEventListener('click', click2);
    console.log('Event #3');
}

function init1() {
    console.log('doc');
    const div = document.querySelector('div');
    const h1 = document.createElement('h1');
    h1.textContent = 'Laurence Svekis';
    h1.onclick = () => {
        console.log('h1 #2');
    }
    h1.addEventListener('click', (e) => {
        console.log('h1 #1');
    })
    h1.onclick = () => {
        console.log('h1 #3');
    }
    h1.addEventListener('click', (e) => {
        console.log('h1 #4');
    })
    h1.addEventListener('click', (e) => {
```

```
        console.log('h1 #5');
    })
    div.append(h1);
    console.log(div);
}


function init() {
    console.log('window');

}
```

Form Select and Input fields values and creation of elements JavaScript Example

Lesson Coding Exercise :

1. To the main page element add a couple of text Nodes. Select them and update the text content of the text nodes.
2. Create an input text field set some common attributes to the input field
3. Create an input field that will be a number type, set the min and max attributes
4. Create an input field set it to date type
5. Create an input field set it to color type
6. Create and add a label, and a select element to the page. Add attributes to select multiple items in the select field
7. Create a number of options for the select field
8. Add a button that will list out all selected options from the select field into an unordered list.
9. Add radio buttons to the page with labels for their values
10. Add a button that will get the value of the selected radio button
11. List out all the input values other than the radio button values

```
const output = document.querySelector('div');
console.log(output);
```

```
output.innerHTML = '';
const out1 = document.createTextNode('Hello');
const out2 = document.createTextNode('World');
output.append(out1);
output.append(out2);
out1.textContent = 'Laurence';
out2.textContent = 'Svekis';
const myInput1 = document.createElement('input');
output.append(myInput1);
myInput1.setAttribute('type','text');
myInput1.setAttribute('name','first');
myInput1.setAttribute('placeholder','FirstName');
myInput1.style.display = 'block';
myInput1.value = 'Laurence';
const myInput2 = document.createElement('input');
output.append(myInput2);
myInput2.setAttribute('type','number');
myInput2.setAttribute('min',1);
myInput2.setAttribute('max',10);
myInput2.value = '5';
myInput2.style.display = 'block';
const myInput3 = document.createElement('input');
output.append(myInput3);
myInput3.setAttribute('type','date');
myInput3.style.display = 'block';
const myInput4 = document.createElement('input');
output.append(myInput4);
myInput4.setAttribute('type','color');
```

```
myInput4.style.display = 'block';

const label1 = document.createElement('label');
label1.setAttribute('for','sel');
label1.textContent = 'My List ';
label1.style.display = 'block';
output.append(label1);
const sel1 = document.createElement('select');
sel1.id = 'sel';
sel1.setAttribute('multiple','');
sel1.setAttribute('size','7');
output.append(sel1);

for(let i=0;i<15;i++){
   const op = document.createElement('option');
   op.textContent = `${i+1} Option`;
   sel1.append(op);
}

const btn = document.createElement('button');
btn.textContent = 'List me';
btn.style.display = 'block';
output.append(btn);

const ul = document.createElement('ul');
output.append(ul);

for(let i=0;i<5;i++){
```

```
    const radio = document.createElement('input');
    radio.setAttribute('type','radio');
    radio.setAttribute('name','radioButtons');
    radio.value = `${i+1} Selection`;
    radio.id = `r${i+1}`;
    const label1 = document.createElement('label');
    label1.textContent = `${i+1} Selection`;
    label1.setAttribute('for', `r${i+1}`);
    output.append(label1);
    output.append(radio);
    if(i==0) radio.setAttribute('checked','');
}
const btn1 = document.createElement('button');
btn1.textContent = 'Get Radios';
btn1.style.display = 'block';
output.append(btn1);
const output1 = document.createElement('div');
output.append(output1);

btn.addEventListener('click',listItems);
btn1.addEventListener('click',getRadios);

function getRadios(){
    const val1 =
document.querySelector('input[name="radioButtons"]:checked');
    console.log(val1.value);
    const ins = document.querySelectorAll('input');
    console.log(ins);
```

```
    ins.forEach(ele =>{
        const getAtt =  ele.getAttribute('type');
        if(getAtt != 'radio'){
            console.log(ele.value)
        };
    })
}


function listItems(){
    const eles = sel1.querySelectorAll('option:checked');
    console.log(eles);
    ul.innerHTML = '';
    eles.forEach(ele =>{
        console.log(ele.textContent);
        const li = document.createElement('li');
        li.style.color = myInput4.value;
        ul.append(li);
        li.textContent = ele.textContent;
    })
}
```

DOM mouse and Keyboard events Page event listeners create Interactions

Lesson Coding Exercise :

1. Create a div and an unordered list to add content into
2. Add a button that will update the style background colors

3. Create an empty array to use as a holding array for page elements you want to update later
4. Create a function that can return a random hex value for the background colors
5. Create 10 divs, set the style properties of the elements, and include a random background color.
6. Add a click event that will set the body background color to match the background of the clicked element
7. Add mouseover, mouseleave, mouseout, mousedown, mouseup, and mouseenter events to all the divs. Create a function that will log the event trigger and a custom message into the output element and into the console.
8. Create 10 input elements, add event listeners onfocus, focusout, onchange to the elements. Update the style properties on the different events
9. Add a keyboard event to track the document. track the keys pressed.
10. Add a keyboard event onkeydown to the element input, if the enter key is pressed add the input field value to the unordered list as a list item
11. Create supporting functions to change the background of the body, random color generator, and a logging function that can output values into the console and into the output element on the page.

```
const div = document.querySelector('div');
div.innerHTML = '';
console.log(div);
const output = document.createElement('div');
div.append(output);
const ul = document.createElement('ul');
div.append(ul);
const btn = document.createElement('button');
btn.textContent = 'Reset Colors';
btn.style.display = 'block';
div.append(btn);
const holder = [];
btn.onclick = (e) => {
   holder.forEach(ele => {
       ele.style.backgroundColor = ranColor();
   })
```

```
}
for (let i = 0; i < 10; i++) {
   const ele = document.createElement('div');
   holder.push(ele);
   ele.style.width = '50px';
   ele.style.height = '50px';
   ele.style.border = '1px solid #ddd';
   ele.style.display = 'inline-block';
   ele.style.backgroundColor = ranColor();
   ele.style.lineHeight = '50px';
   ele.style.textAlign = 'center';
   ele.textContent = `${i+1}`;
   //ele.onclick = updateColor;
   ele.addEventListener('click', () => {
       document.body.style.backgroundColor =
ele.style.backgroundColor;
   });
   ele.onmouseover = (e) => {
       //logger(`mouseOver ${i+1}`);
   }
   ele.addEventListener('mouseover', () => {
       logger(`mouseOver ${i+1}`);
   })
   ele.addEventListener('mouseleave', () => {
       logger(`mouseLeave ${i+1}`);
   })
   ele.addEventListener('mouseout', () => {
       logger(`mouseOut ${i+1}`);
```

```
    })
    ele.addEventListener('mousedown', () => {
        logger(`mouseDown ${i+1}`);
    })
    ele.addEventListener('mouseup', () => {
        logger(`mouseUp ${i+1}`);
    })
    ele.addEventListener('mouseenter', () => {
        logger(`mouseEnter ${i+1}`);
    })
    div.append(ele);
}
for (let i = 0; i < 10; i++) {
    const ele = document.createElement('input');
    ele.style.display = 'block';
    ele.setAttribute('type', 'text');
    ele.value = `${i+1} Input value`;
    div.append(ele);
    ele.onfocus = () => {
        ele.style.color = 'red';
    }
    ele.addEventListener('focusout', () => {
        ele.style.color = 'black';
    });
    ele.onchange = () => {
        ele.style.backgroundColor = 'blue';
    }
    ele.onkeydown = (e) => {
```

```
        if (e.key === 'Enter') {
            adder(ele.value);
            console.log(ele.value);
        }

    }
}
document.addEventListener('keydown', tracker);
document.addEventListener('keyup', tracker);

function adder(val) {
    console.log(val);
    const li = document.createElement('li');
    li.textContent = val;
    ul.append(li);
}

function tracker(e) {
    console.log(e.key);
}

function logger(val) {
    console.log(val);
    output.innerHTML = val;
}

function updateColor() {
    document.body.style.backgroundColor = ranColor();
```

```
}


function ranColor() {
   return `#${Math.random().toString(16).slice(2,8)}`;
}
```

JavaScript RequestAnimationFrame smooth Animations within the DOM Ball bouncing example

The requestAnimationFrame() method can be used to create smooth animations within your web page elements. The browser then calls a specified function that can update an animation before the next repaint. The method takes a callback as an argument to be invoked before the repaint so that you can continue the animation function.

Lesson Coding Exercise :

1. Select the HTMl element that will be used as the parent container
2. Create an element div for displaying the score to the user, create a div that will be the ball in the Lesson Coding Exercise .
3. Update the style properties for the elements, round the ball, position needs to be relative or absolute to be able to move the element on the screen.
4. Create a global object to hold the x,y and the x direction and y direction. Include the speed and other values needed for the game.
5. Assign to a variable the requestAnimationFrame() then create a function that will run the movement.

6. In the animation frame function, check the pos of x and y ensure the values are within the boundaries of the parent, if not change the direction by multiplying the direction value by -1
7. Update the ball positions and apply the new position to the style left and top values.
8. Within the animation function, request the animation frame to keep the animation going.
9. Add an event for keydown values, if enter is pressed cancel the animation, or restart the animation.
10. Add an event listener on the ball, if clicked update the score and reset the position and color of the ball.
11. Create a score update function that will be invoked every time there is a score update.

```
const div = document.querySelector('div');
const score = document.createElement('div');
score.style.textAlign = 'center';
document.body.prepend(score);

const ball = document.createElement('div');
ball.style.border = '3px solid white';
div.innerHTML = '';
div.style.width = '500px';
div.style.height = '400px';
div.style.backgroundColor = 'black';
div.style.margin = 'auto';
div.style.overflow = 'hidden';
const pos = {
   x:100,
   y:100,
   dx:1,
   dy:1,
   speed:5,
   move:true,
   score:0
```

```
}
adderScore();
div.append(ball);
ball.style.backgroundColor = 'red';
ball.style.height = '50px';
ball.style.width = '50px';
ball.style.borderRadius = '50%';
ball.style.position = 'relative';
ball.onclick = gameHit;
let mover = requestAnimationFrame(ani);

document.addEventListener('keydown',(e)=>{
   console.log(e.key);
   if(e.key === 'Enter') {
       if(pos.move){
           pos.move = false;
           cancelAnimationFrame(mover);
       }else{
           pos.move = true;
           mover = requestAnimationFrame(ani);
       }
   }
})
function gameHit(){
   pos.score++;
   ball.style.backgroundColor =
`#${Math.random().toString(16).slice(2,8)}`;
   adderScore();
```

```
    pos.x = Math.floor(Math.random()*450);
    pos.y = Math.floor(Math.random()*350);
    ball.style.left = `${pos.x}px`;
    ball.style.top = `${pos.y}px`;
}

function adderScore(){
    score.innerHTML = `<div>Score :${pos.score}</div>`;
}

function ani(){
    pos.move = true;
    if(pos.move){
        if(pos.x > 450 || pos.x < 0){
            pos.dx *= -1;
        }
        if(pos.y > 350 || pos.y < 0){
            pos.dy *= -1;
        }
        pos.x += pos.speed * pos.dx;
        pos.y += pos.speed * pos.dy;
        ball.style.left = `${pos.x}px`;
        ball.style.top = `${pos.y}px`;
    }
    mover = requestAnimationFrame(ani);
}
```