

JavaScript Projects DOM Interactive Dynamic web pages Introduction web development Course Resource Guide.

By Laurence Svekis

Explore how to create Interactive Dynamic web pages

[JavaScript Projects DOM Interactive Dynamic web pages Introduction web development Course Resource Guide.](#)

[Getting started with JavaScript DOM coding and development](#)

[Web Developer Setup use of Editor for JavaScript Code](#)

[JavaScript Resources to explore more about JavaScript](#)

[JavaScript DOM Object Selecting Page elements and more](#)

[JavaScript querySelectorAll Get Page Elements Select ALL](#)

[Page Events Element Event Listener access page content with JavaScript](#)

[JavaScript and Page Input Values from Page Elements](#)

[How to use JavaScript Request Animation Frame](#)

[JavaScript Starter Projects DOM Simple Projects to Start Coding](#)

[How to make Interactive DOM list saving to localStorage](#)

[JavaScript Component Create a Star Rating Project](#)

[JavaScript Game within the DOM Coin Toss Game Project](#)

[JavaScript Typing Challenge Game with JavaScript DOM](#)

[JavaScript DOM fun with Page Elements Moving Storing Keypress](#)

[JavaScript Combo Guessing Game Exercise](#)

[JavaScript Shape Clicker Game Click the shape quickly to win](#)

[JavaScript Number Guessing Game with Game Logic](#)

[JavaScript DOM Interactive Components and Useful Projects](#)

[Pure JavaScript Accordion hide and show page elements](#)

[JavaScript Drag and Drop Simple Boxes Component](#)

[Dynamic Drag and Drop](#)

[JavaScript Email Extractor Mini Project](#)

[Create a Quiz with Javascript JSON quiz tracker](#)

[JavaScript Image Preview File Reader Example](#)

[JavaScript Interactive Dice Game with Page elements](#)

[JavaScript Dice Game Challenge Lesson](#)

[JavaScript DOM Fun Projects Interactive DOM Elements](#)

[JavaScript Tip Calculator Project](#)

[Tip Calculator Project Part 1](#)

[Tip Calculator Project Part 2](#)

[Pure JavaScript Calculator DOM page elements Project](#)

[JavaScript Calculator Part 1](#)

[JavaScript Calculator Part 2](#)

[JavaScript Calculator Part 3](#)

[JavaScript Bubble Popping DOM Game Coding project](#)

[How to move a Page Element With JavaScript DOM Mover Example](#)

[Collision Detection between Page elements with JavaScript DOM](#)

[JavaScript DOM Interactive Game](#)

Introduction to the DOM and how JavaScript can create page interactions - Explore how to write JavaScript code, select page elements and more. Loaded with useful projects to help you learn more about creating interactive and dynamic web content with JavaScript.

Course covers everything you need to know in a step by step format so that you can start coding JavaScript and creating amazing things with code.

Getting started with JavaScript DOM coding and development

Explore the fundamentals of writing JavaScript code, selecting and manipulating page elements, adding event listeners and more

Web Developer Setup use of Editor for JavaScript Code

Setup of your web development environment, getting started writing code. Use of editor **visual studio code**. Use of Chrome browser and Chrome DevTools to see console messages. Use of `console.log()` to output debugging data to the browser.

<https://code.visualstudio.com/> - Recommended editor used in the course. You can use any editor you are comfortable with to write JavaScript code. HTML files can be run in any browser.

```
<!DOCTYPE html>
<html>
<head><title>JavaScript</title></head>
<body>
  <script src="ex1.js"></script>
</body>
</html>
```

```
console.log('hello world');
```

JavaScript Resources to explore more about JavaScript

Where to get help and more information about JavaScript. Find code examples and where to get help online to learn more about JavaScript and JavaScript DOM. What is the DOM and how it works, the structure of a webpage and what can be accessed via JavaScript.

https://developer.mozilla.org/en-US/docs/Web/API/HTML_DOM_API

JavaScript DOM Object Selecting Page elements and more

How to access page elements, manipulate content and properties of the page element. Learn how to select and update contents of the element. DOM is a tree structure representing all the

page elements and properties of those elements. Introduction to **console.dir()** **document.querySelector()** and **textContent** property value assignment.

```
<!DOCTYPE html>
<html>
<head><title>JavaScript</title></head>
<body>
  <h1>Test</h1>
  <script src="ex1.js"></script>
</body>
</html>
```

```
// console.dir(document);
const myObj = {
  "html" : [
    "head", "body"
  ],
  "h1" : {
    "textContent" : "Hello"
  }
}
//console.log(myObj);

const ele1 = document.querySelector('h1');
ele1.textContent = "Hello World";
//console.log(ele1);
```

JavaScript querySelectorAll Get Page Elements Select ALL

Use of querySelector and querySelectorAll to select matching page elements. Different selectors including tag, id, and class. Select page element, iterate contents of node list output and update page elements.

```
<!DOCTYPE html>
<html>

<head>
  <title>JavaScript</title>
  <style>
    #two {
      background-color: red;
    }
  </style>
</head>

<body>
  <h1>Test</h1>
  <div class="red">One</div>
  <div id="two">Two</div>
  <div class="red">Three</div>
  <script src="ex2.js"></script>
</body>

</html>
```

```
const div1 = document.querySelector('div');
//console.log(div1.textContent);
```

```

const divs = document.querySelectorAll('div');
//console.log(divs);

const ele1 = document.querySelector('.red');
console.log(ele1);
const eles = document.querySelectorAll('.red');
console.log(eles[0]);

ele1.textContent = "Ele1";
eles[0].textContent = "Hello World";
document.body.children[1].textContent = "Via Body";
console.log(ele1.textContent);

const eleID = document.querySelector('#two');
eleID.textContent = "NEW ID";

divs.forEach((div,index)=>{
  div.textContent += ` - (${index})`;
})

```

Page Events Element Event Listener access page content with JavaScript

Make your page elements come to life with event listeners, which listen for events and allow you to run blocks of code once the event gets triggered. Using `addEventListener` method on each element you can add click event listeners. Once clicked output content into the console, then using what you've learned create a click tracker on all your divs on the page. As the elements are clicked update the click count on the element. Try to keep the code short as possible.

Exercise Page Click Tracker:

1. Select the page elements to add click tracker on
2. Update the values of the elements
3. Add a value click tracker property to the element object
4. Get the target element from the event object, use the target event.
5. Increment the counter and update the output of the clicked element.

```
<!DOCTYPE html>
<html>
<head>
  <title>JavaScript</title>
</head>
<body>
  <h1>Test</h1>
  <div class="red">One</div>
  <div id="two">Two</div>
  <div class="red">Three</div>
  <script src="ex3.js"></script>
</body>
</html>
```

```
const eles = document.querySelectorAll('div');
/*
eles[0].addEventListener('click',(e)=>{
  console.log('clicked');
})
eles[1].addEventListener('click',buttonClicker);

eles[1].removeEventListener('click',buttonClicker);
*/
eles.forEach((el)=>{
```

```

    el.addEventListener('click',buttonClicker);
    el.textContent = 'Clicked : 0';
    el.val = 0;
  })

function buttonClicker(e){
  //console.log(e.target);
  const ele = e.target;
  console.log(ele.val);
  ele.val++;
  ele.textContent = `Clicked : ${ele.val}`;
}

```

JavaScript and Page Input Values from Page Elements

This lesson will demonstrate how to get values from input fields, using those values within your JavaScript code. Select the element, get the value and output it to your webpage. Use of document.createElement and append methods to create page elements, and add new page elements to your webpage. Select the page elements and make them interactive, updating the existing text Content of the div with whatever value is contained within the input field.

Exercise : Page message from input value

1. Select the page element that will be the main container.
2. Create new page elements
3. Add new page elements to the container
4. Update content of page elements
5. Add an event listener for the button
6. When the button is clicked update the contents of the div with the value of the contents of the input field.

```

<!DOCTYPE html>
<html>
<head>
  <title>JavaScript</title>
</head>

```



```

<body>
  <div class="output"></div>
  <script src="ex4.js"></script>
</body>
</html>

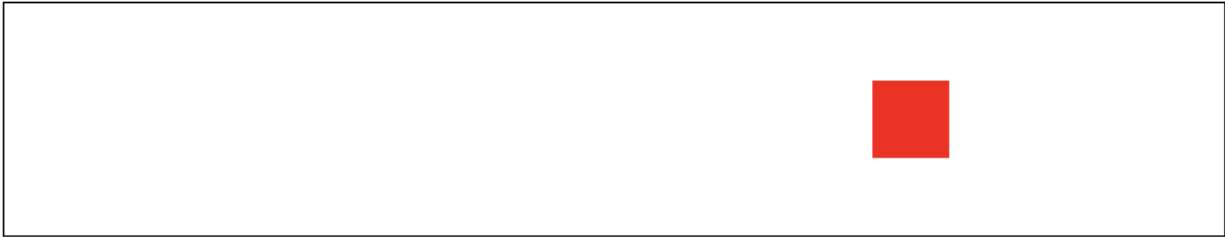
```

```

const output = document.querySelector('.output');
const myMessage = createEle('div', output);
const myInput = createEle('input', output);
const myBtn = createEle('button', output);
//const val1 = output.appendChild(myInput);
//output.append(myInput);
//output.append(myMessage);
myInput.value = "Hello";
myMessage.textContent = "Message";
myBtn.textContent = "Click Me";
myBtn.addEventListener('click', (e) => {
  myMessage.textContent = myInput.value;
  myInput.value = "";
})
function createEle(eleType, eleParent) {
  const ele = document.createElement(eleType);
  eleParent.append(ele);
  return ele;
}

```

How to use JavaScript Request Animation Frame



The `window.requestAnimationFrame()` method tells the browser that you wish to perform an animation. The browser calls a specified function to update an animation before the next repaint. The method takes a callback as an argument to be invoked before the repaint.

```
<!DOCTYPE html>
<html>
<head>
  <title>JavaScript</title>
  <style>
    .output{
      position:absolute;
      left:0px;
      top:50px;
      width:50px;
      height:50px;
      background-color:red;
    }
    .container{
      position:relative;
      width:80%;
      height:150px;
      border:1px solid black;
      margin:auto;
      overflow:hidden;
    }
  </style>
</head>
<body>
  <div class="container">
    <div class="output"></div>
  </div>
  <script src="ex5.js"></script>
</body>
</html>
```

```

const container = document.querySelector('.container');
const output = document.querySelector('.output');
const game = {ani:null,inplay:false,x:0,speed:5};

output.addEventListener('click', (e)=>{
  if(!game.inplay){
    game.ani = window.requestAnimationFrame(mover);
    game.inplay = true;
  }else{
    window.cancelAnimationFrame(game.ani);
    game.inplay = false;
  }
})

function mover(){
  /*    if(game.x > (container.offsetLeft + container.offsetWidth)){
    game.speed *= -1;
  } */
  const dim = container.getBoundingClientRect();
  game.x += game.speed;
  if((game.x > dim.width-50) || (game.x < 0)){
    game.speed *= -1;
  }
  output.style.left = game.x + 'px';
  game.ani = window.requestAnimationFrame(mover);
}

```

JavaScript Starter Projects DOM Simple Projects to Start Coding

Explore the DOM with these simple projects to get you coding and learning more about how to write JavaScript to create page interactions and mini web applications.

How to make Interactive DOM list saving to localStorage

- Svekis
- JavaScript 1000
- Test
- DOM Course
- Laurence
- DOM 5000

Interactive DOM list saving to localStorage. Create a list of items that can be edited and deleted. Store and sync your list items to Local storage so it saves and rebuilds the list when the page reloads.

```
const curList = [];
const output = document.querySelector('.output');
const myInput = createMyElement(output, 'input', 'main');
myInput.setAttribute('type', 'text');
const myBtn = createMyElement(output, 'button', 'btn');
myBtn.textContent = 'Add New User';
const myList = createMyElement(output, 'ul', 'myList');
let getData = localStorage.getItem('curList');
window.addEventListener('DOMContentLoaded', (e) => {
  if (getData) {
    const tempArr = JSON.parse(getData);
```

```

        tempArr.forEach(user =>{
            addNewUser(user);
        })
    }
})

myBtn.addEventListener('click',(e)=>{
    console.log('click');
    let userName = myInput.value;
    if(userName.length > 3){
        const li = addNewUser(userName);
        myInput.value = '';
    }
})

function updater(){
    const myListItems = document.querySelectorAll('.info');
    curList.length = 0;
    myListItems.forEach((el)=>{
        curList.push(el.textContent);
    })
    localStorage.setItem('curList',JSON.stringify(curList));
}

function addNewUser(userName){
    curList.push(userName);
    const li = createMyElement(myList,'li','myList');
    const div = createMyElement(li,'div','container');

```

```

const span1 = createMyElement(div,'span','info');
span1.textContent = userName;
const span2 = createMyElement(div,'span','editor');
span2.textContent = 'Edit';
const span3 = createMyElement(div,'span','del');
span3.textContent = 'Delete';
span2.addEventListener('click',(e)=>{
    if(span2.textContent === 'Edit'){
        span1.style.backgroundColor = 'yellow';
        span1.setAttribute('contenteditable',true);
        span2.textContent = 'Save';

    }else{
        span1.style.backgroundColor = 'white';
        span1.setAttribute('contenteditable',false);
        span2.textContent = 'Edit';
        updater();
    }
})
span3.addEventListener('click',(e)=>{
    li.remove();
    console.log('del');
    updater();
})
updater();
return li;
}

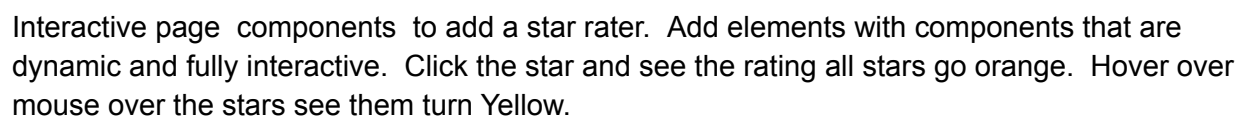
```

```
function createMyElement(parent,elType,classAdd){
    const ele = document.createElement(elType);
    parent.append(ele);
    ele.classList.add(classAdd);
    return ele;
}
```

```
<!DOCTYPE html>
<html>
<head>
  <title>List Builder</title>
  <style>
    *{
      box-sizing:border-box;
    }
    li{
      height:30px;
    }
    .info{
      padding:5px;
    }
    .editor{
      background-color:green;
      max-width:50px;
      padding:2px;
      color:white;
      font-size:0.5em;
      margin:0 2px;
    }

```

```
.del{
    background-color:red;
    max-width:50px;
    padding:2px;
    color:white;
    font-size:0.5em;
    margin:0 2px;
}
</style>
</head>
<body>
    <div class="output"></div>
    <script src="list.js"></script>
</body>
</html>
```

```
<!DOCTYPE html>

<html>

<head>

  <title>Star Rater</title>

  <style>

    .stars ul{

      list-style-type: none;

      padding:0;

    }

  }

</style>

</head>

</html>
```

```
.star{
    font-size:2em;
    color:#ddd;
    display:inline-block;
}

.yellow{
    color:yellow;
}

.orange{
    color:orange;
}

.output{
    background-color:#ddd;
    border:1px solid #eee;
    margin:10px 0;
    padding:10px;
}

</style>
</head>
<body>
    <div class="stars"></div>
    <div class="stars"></div>
    <div class="stars"></div>
    <div class="stars"></div>
    <script src="star.js"></script>
</body>
</html>
```

```

const starContainers = document.querySelectorAll('.stars');
starContainers.forEach((el)=>{
  const starsUL = createElements(el,'ul','main');
  const output = createElements(el,'div','output');
  for(let x=0;x<5;x++){
    const star = createElements(starsUL , 'li','star');
    star.innerHTML = '★';
    star.starValue = (x+1);
    ["mouseover","mouseout","click"].forEach((ele)=>{
      star.addEventListener(ele,starRate);
    })
  }
})

function starRate(e){
  //console.log(e.type);
  //console.log(e.target.starValue);
  const eventType = e.type;
  const parent = e.target.closest('.stars');
  console.log(parent);
  const output = parent.querySelector('.output');
  const curStars = parent.querySelectorAll('.star');
  if(eventType === 'click'){
    output.innerHTML = `You Rated this ${e.target.starValue}
stars`;
    addColor(curStars,e.target.starValue);
  }else if(eventType === 'mouseover'){
    addYellow(curStars,e.target.starValue);
  }
}

```

```
}  
}  
function addYellow(curStars,val){  
  console.log(val);  
  curStars.forEach((star,index)=>{  
    if(index < val){  
      star.classList.add('yellow');  
    }else{  
      star.classList.remove('yellow');  
    }  
  })  
}  
  
function addColor(curStars,val){  
  console.log(val);  
  curStars.forEach((star,index)=>{  
    if(index < val){  
      star.classList.add('orange');  
    }else{  
      star.classList.remove('orange');  
    }  
  })  
}  
  
function createElements(parent,elType,myClass){  
  const el = document.createElement(elType);  
  el.classList.add(myClass);  
  parent.append(el);  
}
```

```
return el;
}
```

JavaScript Game within the DOM Coin Toss Game Project



Mini Game for player to guess the result of a coin toss, player selects either red or blue to guess the result of the random coin toss. Tracks streak and score if guessed correctly. Project to explore random and use of logic for game design with JavaScript.

```
<!DOCTYPE html>
<html>
<head>
  <title>Coin Toss</title>
  <style>
    .coin{
      font-size:6em;
      width:100px;
      height:100px;
```

```
        position: relative;
        text-align: center;
        margin: auto;
    }
    .coin div{
        position: absolute;
        width: 100%;
        height: 100%;
        background-color: #eee;
        border-radius: 50%;
        line-height: 100px;
        text-align: center;
        color: white;
    }
    .coin div:first-child{
        background-color: red;
    }
    .coin div:last-child{
        background-color: blue;
    }
    .btn{
        width: 100%;
        display: inline-block;
        margin: 5px;
        font-size: 1.5em;
        color: white;
background-color: black;
    }
```

```
    .score{
        font-family:Impact, Haettenschweiler, 'Arial Narrow
Bold', sans-serif;
        font-size:1.5em;
        text-align:center;
    }
    .message{
        font-size:1em;
        background:#ddd;
        text-align:center;
        padding:10px;
    }
    .main{
        margin:auto;
        width:300px;
        background-color:#eee;
    }
</style>
</head>
<body>
    <div class="main"></div>
    <script src="coin.js"></script>
</body>
</html>
```

```
const main = document.querySelector('.main');
const game = {score:0,streak:0};
```

```
const btn = document.createElement('button');
const btn1 = document.createElement('button');
const output = document.createElement('div');
const scoring = document.createElement('div');
const message = document.createElement('div');
message.classList.add('message');
message.textContent = 'Press Button to Start Game';
main.append(message);
main.append(scoring);
main.append(output);
main.append(btn);
main.append(btn1);
btn.classList.add('btn');
btn1.classList.add('btn');
scoring.classList.add('score');
const coin = document.createElement('div');
const sidea = document.createElement('div');
const sideb = document.createElement('div');
output.append(coin);
coin.append(sidea);
coin.append(sideb);

sidea.innerHTML = "&#9786;";
sideb.innerHTML = "&#9785;";
coin.classList.add('coin');
coin.style.display = 'none';
btn.textContent = "Start Game";
btn.addEventListener('click',playGame);
```



```
btn1.textContent = "Heads";
btn1.addEventListener('click',playGame);
btn1.style.display = 'none';

btn.style.backgroundColor = 'red';
btn1.style.backgroundColor = 'blue';

function playGame(e){
    console.log(e.target.textContent);
    const el = e.target;
    if(el.textContent === 'Start Game'){
        game.score = 0;
        game.streak = 0;
        message.textContent = 'Select either Heads or Tails';
        btn.textContent = "Heads";
        btn1.textContent = "Tails";
        btn1.style.display = 'block';
        coin.style.display = 'block';
        addScore();
    }else if(el.textContent === 'Heads'){
        //console.log('flip coin');
        coinFlipper('Heads');
    }else if(el.textContent === 'Tails'){
        //console.log('flip coin');
        coinFlipper('Tails');
```

```

    }

    //btn.style.display = 'none';
}

function coinFlipper(val){
    const ranValue = Math.floor(Math.random()*2);
    let result = "";
    if(ranValue === 1){
        sidea.style.display = 'block';
        sideb.style.display = 'none';
        result = "Heads";
    }else{
        sideb.style.display = 'block';
        sidea.style.display = 'none';
        result = "Tails";
    }
    //add scoring
    if(result === val){
        game.score++;
        game.streak++;
        message.textContent = `You Picked ${val} Correct it was
${result}`;
    }else{
        game.streak=0;
        game.score--;
        message.textContent = `You Picked ${val} Wrong!!! was
${result}`;
    }
}

```

```

    addScore();
    return result;
}

function addScore(){
    scoring.innerHTML = `Score : ${game.score} Streak
    (${game.streak})`;
}

```

JavaScript Typing Challenge Game with JavaScript DOM

Time:7.728 Score 2 out of 4

Do You like Java

Start

Create a typing game that counts the seconds it takes to type the word phrase that is displayed. Code will check for word accuracy and time it takes displaying the results to the player.

```

<!DOCTYPE html>
<html>
<head>
    <title></title>
    <style>
        .typer{
            width:100%;

```

```
        height:50px;
        text-align:center;
        font-size:2em;
    }
    .btn{
        width:100%;
        font-size:2em;
        padding:10px;
        background-color:black;
        color:white;
    }
    .main{
        text-align:center;
        padding:10px;
        font-size:1.5em;
        border:3px solid white;
    }
</style>
</head>
<body>
    <div class="main"></div>
    <textarea name="words" class="typer"></textarea>
    <br>
    <button class="btn">Start</button>
    <script src="typer.js"></script>
</body>
</html>
```

```
const main = document.querySelector('.main');
const typeArea = document.querySelector('.typer');
const btn = document.querySelector('.btn');
const wording = ["Do you like JavaScript", "Have fun with Code"];
const game = {start:0, end:0, user:"", arrText:""};
typeArea.disabled = true;
btn.addEventListener('click', (e) => {
    if (btn.textContent === 'Start') {
        playGame();
        typeArea.value = "";
        typeArea.disabled = false;
    } else if (btn.textContent === 'Done') {
        typeArea.disabled = true;

        main.style.borderColor = 'white';
        endPlay();
    }
})

function endPlay() {
    const date = new Date();
    game.end = date.getTime();
    const totalTime = ((game.end - game.start) / 1000);
    game.user = typeArea.value;
    const correct = checkResult();
    console.log(correct);
    main.style.borderColor = 'white';
}
```

```
    main.innerHTML = `Time:${totalTime} Score ${correct.score}
out of ${correct.total}`;
    btn.textContent = 'Start';
}

function checkResult(){
    let val1 = game.arrText.split(" ");
    let val2 = game.user.split(" ");
    let score = 0;
    val1.forEach((word,index)=>{
        if(word === val2[index]){
            score++;
        }
    })
    return {score:score,total:val1.length}
}

function playGame(){
    let ranText = Math.floor(Math.random()*wording.length);
    main.textContent = wording[ranText];
    game.arrText = wording[ranText];
    main.style.borderColor = 'red';
    btn.textContent = 'Done';
    const date = new Date();
    game.start = date.getTime();
}
```

JavaScript DOM fun with Page Elements Moving Storing Keypress



Simple application to store keypress and move an element on the page. Explore moving and manipulating page elements with JavaScript. Listen for keyboard presses and apply the action accordingly. The application will store the keypresses and then launch the movement once the enter key is pressed. Excellent resource to learn more about keyboard events and how to track movement and enable element movement on the web page.

```
<!DOCTYPE html>
<html>
<head>
  <title>Element Path Mover</title>
  <style>
    .ele{
      position:absolute;
      width:50px;
      height:50px;
      background-color:red;
      text-align:center;
```

```
        top:0px;
        left:0px;
    }
    .main{
        position:relative;
        border:1px solid black;
        width:100%;
        height:500px;
    }
    .box{
        padding:4px;
        border:1px solid #eee;
        font-size:1.2em;
        color:white;
        margin:0 1px;
    }
    .output{
        width:100%;
        overflow:scroll;
    }
    .up{
        background-color:red;
    }
    .down{
        background-color:blue;
    }
    .right{
        background-color:green;
```



```

    }
    .left{
        background-color:black;
    }
</style>
</head>
<body>
    <div class="output"></div>
    <div class="main"></div>

    <script src="mover.js"></script>
</body>
</html>

```

```

const main = document.querySelector('.main');
const output = document.querySelector('.output');
const ele = document.createElement('div');
const game = {actions:[]};
main.append(ele);
ele.classList.add('ele');

document.addEventListener('keydown',(e)=>{
    e.preventDefault();
    console.log(e.key);
    if(e.key === 'ArrowLeft'){
        addAction('left');
    } else if(e.key === 'ArrowRight'){
        addAction('right');
    }
});

```

```

    } else if(e.key === 'ArrowDown'){
        addAction('down');
    } else if(e.key === 'ArrowUp'){
        addAction('up');
    } else if(e.key === 'Enter'){
        mover();
/*
        while(game.actions.length){
            const ele = game.actions.shift();
            const val = ele.textContent;
            ele.remove();
        } */
    }

}))

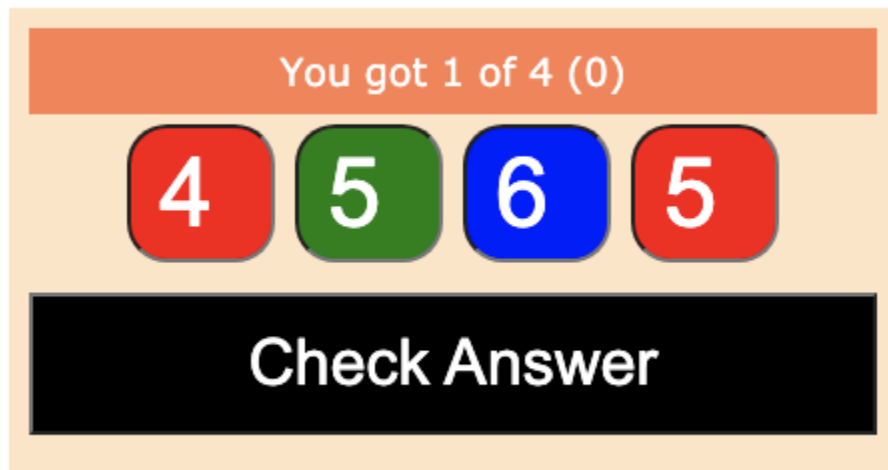
function mover(){
    if(game.actions.length > 0){
        const ele1 = game.actions.shift();
        const val = ele1.textContent;
        //const cur = ele.getBoundingClientRect();
        ele1.remove();
        //console.log(cur);
        const pos = [ele.offsetLeft,ele.offsetTop];
        if(val==="up"){
            pos[1]-=50;
        } else if(val==="down"){
            pos[1]+=50;

```

```
    } else if(val==="left"){
        pos[0]-=50;
    } else if(val==="right"){
        pos[0]+=50;
    }
    ele.style.left = pos[0] + 'px';
    ele.style.top = pos[1] + 'px';
    setTimeout(mover,300);
}
return;
}

function addAction(val){
    const span = document.createElement('span');
    span.textContent = val;
    span.classList.add('box');
    span.classList.add(val);
    game.actions.push(span);
    output.append(span);
    //console.log(game.actions);
}
```

JavaScript Combo Guessing Game Exercise



Guess the combo of the random numbers that are hidden within the dials. Red means you guessed too low, red means you guessed too high green means you got it correct. Apply logic, get the inputs and provide the user feedback as a response. Interactive game which can be expanded to have more numbers for guessing the values that are hidden in each. Great way to learn more about JavaScript and how you can apply JavaScript for a simple interactive game.

```
<!DOCTYPE html>
<html>
<head>
  <title>Combo Game</title>
  <style>
    .dial{
      width:60px;
      border-radius:20px;
      text-align:center;
      padding:5px;
      font-size:3em;
      margin:5px;
    }
    .btn{
```

```
        width:100%;
        padding:15px;
        font-size:2em;
        background-color:black;
        color:white;
        margin:10px 0;
    }
    .output{
        width:80%;
        margin:auto;
        text-align:center;
        padding:10px;
        background-color:bisque;
    }
    .message{
        background-color:coral;
        color:white;
        font-family:Verdana, Geneva, Tahoma, sans-serif;
        padding:10px;
        font-size:1.2em;
    }
</style>
</head>
<body>
    <div class="output"></div>
    <script src="combo.js"></script>
</body>
</html>
```

```

const output = document.querySelector('.output');
const message = document.createElement('div');
const gameArea = document.createElement('div');
const btn = document.createElement('button');
const game = {guesses:0,num:4};
output.append(message);
output.append(gameArea);
output.append(btn);
btn.classList.add('btn');
message.classList.add('message');
btn.textContent = 'Start Game';
outputMessage('Click button to start game');

btn.addEventListener('click',(e)=>{
  if(btn.textContent === 'Start Game'){
    game.guesses = 0;
    makeBoard();
    btn.textContent = 'Check Answer';
    outputMessage('Guess the combo adjust the dials');
  } else if(btn.textContent === 'Check Answer'){
    checkAnswer();
    game.guesses++;
  }
})

function checkAnswer(){
  const combos = document.querySelectorAll('.dial');

```

```

let winners = 0;
combos.forEach((el)=>{
    //console.log(el.correct);
    el.style.color = 'white';
    if(el.correct == el.value){
        winners++;
        el.style.backgroundColor = 'green';
    } else if(el.correct > el.value){
        el.style.backgroundColor = 'red';
    } else if(el.correct < el.value){
        el.style.backgroundColor = 'blue';
    }
})
if(winners === combos.length){

    gameOver();
} else {
    outputMessage(`You got ${winners} of ${combos.length}
(${game.guesses})`);
}
}

function gameOver(){
    outputMessage(`Game Over it took ${game.guesses} guesses`);
    btn.textContent = 'Start Game';
}

function makeBoard(){

```

```

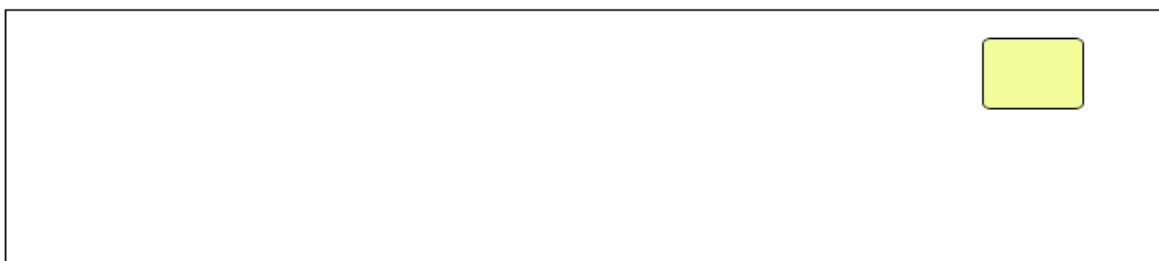
gameArea.innerHTML = "";
for(let x=0;x<game.num;x++){
    const ele = document.createElement('input');
    ele.setAttribute('type','number');
    ele.max = 9;
    ele.min = 0;
    ele.correct = Math.floor(Math.random()*10);
    ele.classList.add('dial');
    ele.value = 5;
    gameArea.append(ele);
}
}

function outputMessage(html){
    message.innerHTML = html;
}

```

JavaScript Shape Clicker Game Click the shape quickly to win

It took 1.11 seconds to click



Click the shape as quickly as possible, it tracks the time it takes in seconds to click it. This lesson will demonstrate how to move page elements, how to track time with a timer within JavaScript. Check and show scoring to the player. Continuous gameplay with ability to restart.


```
<!DOCTYPE html>
<html>
<head>
  <title>Click Me Game</title>
  <style>
    .output{
      width:80%;
      height:500px;
      border:1px solid black;
      margin:auto;
      text-align:center;
    }
    .box{
      width:60%;
      height:100px;
      position:relative;
      top:50px;
      left:20%;
      background-color:cornsilk;
      border:1px solid black;
      font-size:1.5em;
    }
    .message{
      text-align:center;
      padding:10px;
      font-size:1.3em;
    }
  </style>
```

```

</head>
<body>
  <div class="output"></div>
  <script src="clickme.js"></script>
</body>
</html>

```

```

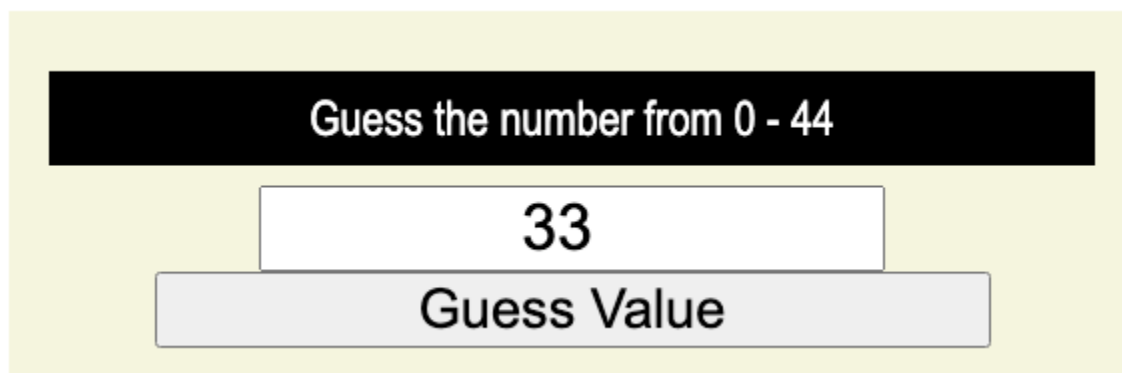
const output = document.querySelector('.output');
const message = document.createElement('div');
message.classList.add('message');
document.body.prepend(message);
message.textContent = "Press to Start";
const box = document.createElement('div');
const game = {timer:0,start:null};
box.classList.add('box');
box.textContent = 'Click to Start Game';
output.append(box);

box.addEventListener('click',(e)=>{
  box.textContent = "";
  box.style.display = 'none';
  game.timer = setTimeout(addBox,ranNum(3000));
  if(!game.start){
    message.textContent = 'Watch for element and click it';
  }else{
    const cur = new Date().getTime();
    const dur = (cur - game.start)/1000;
    message.textContent = `It took ${dur} seconds to click`;
  }
});

```

```
}  
}))  
  
function addBox(){  
    game.start = new Date().getTime();  
    const container = output.getBoundingClientRect();  
    console.log(container);  
    const dim = [ranNum(50) + 20, ranNum(50) + 20];  
    box.style.display = 'block';  
    box.style.width = dim[0] + 'px';  
    box.style.height = dim[1] + 'px';  
    box.style.backgroundColor = '#' +  
Math.random().toString(16).substr(-6);  
    box.style.left = ranNum(container.width - dim[0]) + 'px';  
    box.style.top = ranNum(container.height - dim[1]) + 'px';  
    box.style.borderRadius = ranNum(50) + '%';  
    //box.style.left = ranNum();  
}  
  
function ranNum(max){  
    return Math.floor(Math.random()*max);  
}
```

JavaScript Number Guessing Game with Game Logic



Dynamic mini game asking the player to guess the hidden number. Tracks the number of guesses as a score. Enter a guess, get feedback if it's too high or low and then guess again. Try to solve the hidden number in as little guesses as possible.

```
<!DOCTYPE html>
<html>
<head>
  <title>Number Guess</title>
  <style>
    *{
      box-sizing:border-box;
    }
    .output{
      width:90%;
      margin:auto;
      background-color:beige;
      text-align:center;
      padding:20px;
      font-family:'Franklin Gothic Medium', 'Arial Narrow',
      Arial, sans-serif
    }
  </style>
</head>
<body>
  <div class="output">
    <div>Guess the number from 0 - 44</div>
    <div>33</div>
    <div>Guess Value</div>
  </div>
</body>
</html>
```

```
.main{
  padding:10px;
  background-color:black;
  color:white;
  font-size:1.5em;
  margin:10px 0;
}

.btn{
  margin:10px;
  width:80%;
  margin:auto;
  display:block;
  font-size:1.7em;
}

.guess{
  width:60%;
  font-size:2em;
  margin:auto;
  display:block;
  text-align:center;
}

</style>
</head>
<body>
  <div class="output"></div>
  <script src="guess.js"></script>
</body>
</html>
```

```

const output = document.querySelector('.output');
const main = maker('div',output,'main','Press Button to Start');
const guess = maker('input',output,'guess','');
const btn = maker('button',output,'btn','Start Game');
const game = {hiddenNum : 0,inplay:false,max:10,score:0};
guess.setAttribute('type','number');
guess.style.display = 'none';

btn.addEventListener('click',()=>{
  if(!game.inplay){
    game.num = genNumber(game.max);
    game.inplay = true;
    game.score = 0;
    btn.textContent = "Guess Value";
    guess.style.display = 'block';
    message(`Guess the number from 0 - ${game.max}`, 'black')
  }else{
    game.score++;
    let val = Number(guess.value);
    guess.value="";
    if(val === game.num){
      message(`Your Guess ${game.num} is Correct!<br>Total
Guesses : (${game.score}) `, 'green');
      endGame();
    } else if(val > game.num){
      message(`${val} was too high `, 'red')
    } else if(val < game.num){
      message(`${val} was too low `, 'blue')
    }
  }
});

```

```
    }  
  }  
}))  
  
function endGame(){  
  btn.textContent = "Restart Game";  
  game.inplay = false;  
  guess.style.display = 'none';  
  game.max=genNumber(100);  
}  
  
function message(html,txColor){  
  main.innerHTML = html;  
  main.style.backgroundColor = txColor;  
}  
  
function genNumber(max){  
  return Math.floor(Math.random()*max +1);  
}  
  
function maker(eleType,elParent,eleClass,html){  
  const ele= document.createElement(eleType);  
  ele.classList.add(eleClass);  
  ele.innerHTML = html;
```

```
elParent.append(ele);  
return ele;  
}
```

JavaScript DOM Interactive Components and Useful Projects

Explore using JavaScript with DOM interaction to create useful projects that will help demonstrate using the DOM and JavaScript. Explore element manipulation, use of common JavaScript methods and applying logic to create mini applications.

JavaScript Accordion #1

JavaScript Accordion #2

JavaScript Accordion #3

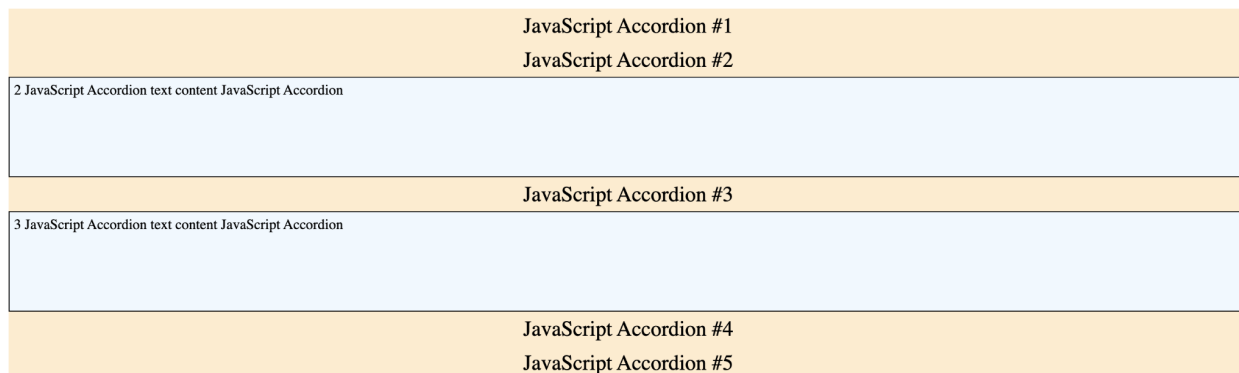
3 JavaScript Accordion text content JavaScript Accordion

JavaScript Accordion #4

JavaScript Accordion #5

5 JavaScript Accordion text content JavaScript Accordion

Pure JavaScript Accordion hide and show page elements



Using just JavaScript, create an Accordion component. Explore how to hide and show page elements with JavaScript, select elements while traversing the DOM tree and apply style properties dynamically all with JavaScript code.

```
<!DOCTYPE html>
<html>
<head>
  <title>JavaScript Accordion</title>
  <style>
    .myText {
      display: none;
      width:100%;
      border: 1px solid black;
      background-color: aliceblue;
      min-height: 100px;
      padding: 5px;
    }
    .title{
      text-align: center;
      font-size: 1.5em;
      background-color: blanchedalmond;
    }
  </style>
</head>
<body>
  <div>
    <div>JavaScript Accordion #1</div>
    <div>JavaScript Accordion #2</div>
    <div>JavaScript Accordion #3</div>
    <div>JavaScript Accordion #4</div>
    <div>JavaScript Accordion #5</div>
  </div>
</body>
</html>
```

```

        padding: 5px;
    }
    .active {
        display: block !important;
    }
</style>
</head>
<body>
    <div class="output">
        <div class="title">JavaScript Accordion #1</div>
        <div class="myText">1 JavaScript Accordion text content
JavaScript Accordion</div>
        <div class="title">JavaScript Accordion #2</div>
        <div class="myText">2 JavaScript Accordion text content
JavaScript Accordion</div>
        <div class="title">JavaScript Accordion #3</div>
        <div class="myText">3 JavaScript Accordion text content
JavaScript Accordion</div>
        <div class="title">JavaScript Accordion #4</div>
        <div class="myText">4 JavaScript Accordion text content
JavaScript Accordion</div>
        <div class="title">JavaScript Accordion #5</div>
        <div class="myText">5 JavaScript Accordion text content
JavaScript Accordion</div>
    </div>
    <script src="accordion.js"></script>
</body>

```

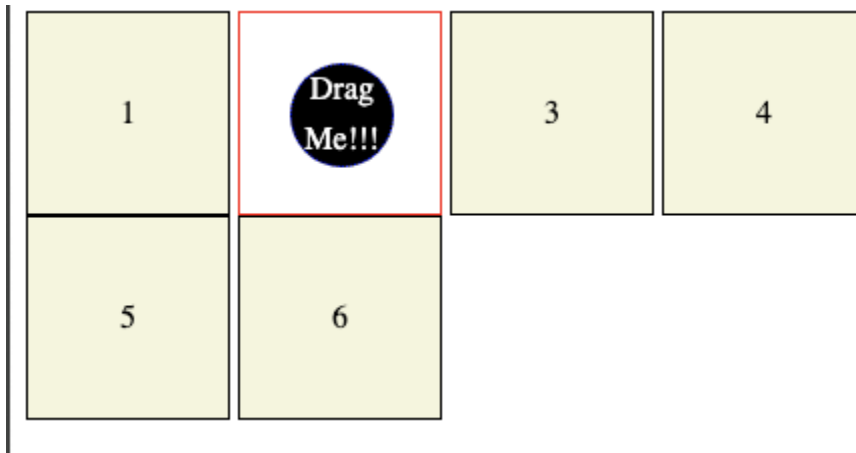
```
</html>
```

```
const titles = document.querySelectorAll('.title');
const textDiv = document.querySelectorAll('.myText');

titles.forEach((el)=>{
  el.addEventListener('click',(e)=>{
    console.log(el.nextElementSibling);
    //removeActive();
    el.nextElementSibling.classList.toggle('active');
  })
})

function removeActive(){
  const activeEles = document.querySelectorAll('.active');
  activeEles.forEach((el)=>{
    el.classList.remove('active');
  })
}
```

JavaScript Drag and Drop Simple Boxes Component



Create a simple drag and drop into page elements with event listeners. Apply CSS for styling. Commonly used functionality which can be done with JavaScript to move page elements from one location to another. Move the element into another parent with drag and drop. Creating multiple page elements that the draggable item can be placed in.

```
<!DOCTYPE html>
<html>
<head>
  <title>JavaScript Drag Drop App</title>
  <style>
    .box{
      width:100px;
      height:100px;
      border:1px solid black;
      background-color: beige;
      text-align: center;
      line-height: 100px;
      position: relative;
      display: inline-block;
    }
    .red{
```

```

        background-color: white;
        border-color: red;

    }

    #dragger{
        position: absolute;
        top:25px;
        left:25px;
        border-radius: 40px;
        border: 1px dotted blue;
        width:50px;
        height:50px;
        line-height: 25px;
        background-color: black;
        color:white;
        font-size: 1em;
    }
</style>
</head>
<body>
    <div class="output">
        <div class="box">1
            <div id="dragger" draggable="true">
                Drag Me!!!
            </div>
        </div>
        <div class="box">2</div>
        <div class="box">3</div>
    </div>

```

```

    <div class="box">4</div>
    <div class="box">5</div>
    <div class="box">6</div>
  </div>
  <script src="dragger.js"></script>
</body>
</html>

```

```

const dragme = document.querySelector('#dragger');
const boxes = document.querySelectorAll('.box');

dragme.addEventListener('dragstart', (e) => {
  dragme.style.opacity = .5;
})

dragme.addEventListener('dragend', (e) => {
  dragme.style.opacity = "";
})

boxes.forEach((box) => {
  box.addEventListener('dragenter', (e) => {
    e.target.classList.add('red');
  })
  box.addEventListener('dragleave', (e) => {
    e.target.classList.remove('red');
  })
  box.addEventListener('dragover', (e) => {
    e.preventDefault();
  })
  box.addEventListener('drop', (e) => {

```

```

    e.preventDefault();
    console.log('dropped');
    e.target.appendChild(dragme);
  })
})

```

Dynamic Drag and Drop



Create a fully dynamic Grid of cells, add new draggable items to the container. All items can be selected and dragged and dropped to any cell within the container. Extend on the drag and drop by adding a dynamic container with elements that can be dragged that are dynamically generated and added to the page using JavaScript code.

```

<!DOCTYPE html>
<html>
<head>
  <title>JavaScript Dynamic Drag and Drop </title>
  <style>
    * {
      box-sizing: border-box;
    }
  </style>

```

```
        margin:0;
        padding:0;
    }
    .container{
        height:100vh;
        flex-direction : column;
        align-items:center;
        justify-content:center;
    }
    .output{
        display:grid;
        align-items:center;
    }
    .box{
        min-height:80px;
        border:1px solid #ccc;
        display:inline-flex;
        align-items:center;
        justify-content:center;
    }
    .drag-over{
        border:dashed 4px green;
    }
    .hide{
        display:none;
    }
    .item{
        cursor:move;
```



```

        padding:10px;
        width:80px;
        height:80px;
        text-align:center;
    }
</style>
</head>
<body>
    <div class="container">
        <div><button>Add One</button></div>
        <div class="output"></div>
    </div>
    <script src="dyno.js"></script>
</body>
</html>

```

```

const g = {
    items:0,
    size:4,
    id :{},
    box:{}
};

let output = document.querySelector('.output');
const btn = document.querySelector('button');
btn.addEventListener('click', addDropper);

document.addEventListener('DOMContentLoaded',()=>{
    console.log('ready');
    output.style.gridTemplateColumns = `repeat(${g.size}, 1fr)`;

```

```

output.style.gridTemplateRows = `repeat(${g.size}, 1fr)`;
let cell = 0;
for(let x=0;x<g.size;x++){
    for(let y=0;y<g.size;y++){
        const el = createSquare();
        el.addEventListener('dragenter', dragEnter);
        el.addEventListener('dragover', dragOver);
        el.addEventListener('dragleave', dragLeave);
        el.addEventListener('drop', dragDrop);
    }
}
addDropper();
})

function createSquare(){
    const div = document.createElement('div');
    div.classList.add('box');
    output.append(div);
    return div;
}

function addDropper(){
    g.items++;
    const div = document.createElement('div');
    const first = output.querySelector('.box');
    first.append(div);
    let ranColor = Math.random().toString(16).substr(-6);
    div.style.backgroundColor = '#' + ranColor;

```

```
div.textContent = 'ID ' + g.items;
div.setAttribute('id','id'+g.items);
div.classList.add('item');
div.setAttribute('draggable',true);
div.addEventListener('dragstart', dragStart);
}

function dragStart(e){
    g.id = e.target.id;
    console.log(g.id);
}

function dragEnter(e){
    e.preventDefault();
    g.box = e.target.closest('.box');
    e.target.classList.add('drag-over');
    console.log(g.id);
}

function dragOver(e){
    e.preventDefault();
    moveBox();
    e.target.classList.add('drag-over');
    console.log(g.id);
}

function dragLeave(e){
    console.log(g.id);
    moveBox();
    e.target.classList.remove('drag-over');
```

```

}

function dragDrop(e){
    console.log(g.id);
    e.target.classList.remove('drag-over');
    g.box = e.target.closest('.box');
    moveBox();
}

function moveBox(){
    const draggedEle = document.getElementById(g.id);
    const temp = g.box.appendChild(draggedEle);
    console.log(temp);
    return draggedEle;
}

```

JavaScript Email Extractor Mini Project

; ▼

Emails Found : 2

svekis@gmail.com;svekis1@gmail.com

Create your own email extractor tool with pure JavaScript code. Use of regex and JavaScript logic to create and select email addresses from text blocks. Extract all the email addresses from a block of text, get unique addresses as a response output.

<https://regexr.com/> - Useful resource for Regex expressions

```

<!DOCTYPE html>

<html>

```

```

<head>
  <title>JavaScript </title>
  <style>
    .textContent{
      width:100%;
      height:200px;
    }
    .results{
      background-color:bisque;
      border:1px solid black;
      width:100%;
    }
  </style>
</head>
<body>
  <div class="output">
    <textarea name="txtarea" class="textContent">dj askajsd
kjksad svekis@gmail.com laurence svekis1@gmail.com</textarea>
    <br><button>Get Email</button>
    <select><option value=";">;</option>
      <option value=",">,</option>
      <option value=" "> </option></select>
    <div class="results"></div>
  </div>
  <script src="email.js"></script>
</body>
</html>

```

```

const rawText =
document.querySelector('textarea[name="txtarea"]');
const btn = document.querySelector('button');
const output = document.querySelector('.results');
const sel = document.querySelector('select');
btn.addEventListener('click', ()=>{
    console.log(sel.value);
    let temp = rawText.value;
    let exp =
/([A-Za-z0-9._-]+@[A-Za-z0-9._-]+\.[A-Za-z0-9._-]+)/gi;
    let emails = temp.match(exp);
    let holder = [];
    emails.forEach((email)=>{
        console.log(email);
        if(!holder.includes(email)){
            holder.push(email);
        }
    })
    console.log(holder);
    let tempHolder = holder.join(sel.value);
    console.log(`Emails Found ${holder.length}`);
    output.innerHTML = `Emails Found : ${holder.length}`;
    const div = document.createElement('div');
    output.append(div);
    const newResults = document.createElement('textarea');
    newResults.style.width = '100%';
    div.append(newResults);
    newResults.value = tempHolder;

```

```
newResults.addEventListener('click',()=>{  
    newResults.select();  
})  
})
```

Create a Quiz with Javascript JSON quiz tracker

Question : 3 of 3

Score : 2

What color is Grass?

Blue	Green	Red	Purple
------	-------	-----	--------

Wrong

Next Question

Create your own JavaScript based quiz with dynamic quiz pages, scoring and response tracking. Explore how to generate a quiz from JSON data using JavaScript to output the content. Interactive and dynamic quiz game with JavaScript. Create a Quiz with Javascript JSON quiz tracker. How to Create a quiz with JavaScript using JSON data. Dynamically generating a quiz from JSON.

```
<!DOCTYPE html>  
<html>  
<head>  
    <title>JavaScript Quiz</title>  
    <style>  
        .inactive{  
            color:#ddd;  
            background:white;
```

```
        cursor:none;
    }
    .correct{
        color:black;
    }
    .btn{
        display:inline-block;
        padding:8px;
        border-radius:20px;
        background-color:black;
        color:white;
        text-align:center;
        margin:auto;
        margin-top:25px;

    }
    .opt{
        border:1px solid #ccc;
        padding:10px;
        display:inline-block;
    }
    .ques{
        display:block;
        font-size:2em;
        margin-bottom:25px;
    }
    .message{
        text-align:center;
```



```

        padding:10px;
    }
</style>
</head>
<body>
    <div class="output"></div>
    <div class="question"></div>
    <div class="controls"></div>
    <script src="quiz.js"></script>
</body>
</html>

```

```

const question = document.querySelector('.question');
const contrs = document.querySelector('.controls');
const output = document.querySelector('.output');
const game = {
    data: {},
    score: 0,
    questions: 0,
    cur: 0
};
contrs.classList.add('btn');
const url = 'quiz.json';
contrs.addEventListener('click', nextQuestion);
document.addEventListener('DOMContentLoaded', () => {
    startGame();
})

function startGame() {

```

```

    game.score = 0;
    game.cur = 0;
    contrs.style.display = 'block';
    contrs.textContent = "start game click here";
    fetch(url).then((res) => res.json()).then((data) => {
        console.log(data);
        game.data = data;
        game.questions = data.length;
        output.innerHTML = `Total Questions : ${game.questions}`;
    })
}

function scoreboard() {
    output.innerHTML = `Question : ${game.cur} of
${game.questions}`;
    output.innerHTML += `

```

```

    question.innerHTML += `<div>${game.score} out of
    ${game.questions}</div>`;
    const btnStart = document.createElement('button');
    question.append(btnStart);
    btnStart.textContent = "Start new Game";
    btnStart.addEventListener('click', (e) => {
        btnStart.style.display = 'none';
        startGame();
    });
}

function nextQuestion() {
    contrs.style.display = 'none';
    if (game.data.length > 0) {
        game.cur++;
        question.innerHTML = "";
        const temp = game.data.pop();
        console.log(temp);
        const div1 = document.createElement('div');
        div1.textContent = `${temp.question}?`;
        div1.classList.add('ques');
        question.append(div1);
        const div = document.createElement('div');
        div.classList.add('main');
        div.ans = temp.ans;
        temp.opts.push(temp.ans);
        const opts = temp.opts;
        console.log(opts);
    }
}

```

```

    randomArray(opts);
    opts.forEach((sel) => {
        let span = document.createElement('span');
        span.classList.add('opt');
        span.textContent = sel;
        div.append(span);
        span.addEventListener('click', checker);
    })
    question.append(div);

} else {
    console.log('game over');
    gameOver();
}
scoreboard();
}

function checker(e) {
    let sel = e.target;
    let div = sel.closest('.main');
    const eles = document.querySelectorAll('.opt');
    eles.forEach((el) => {
        el.classList.add('inactive');
        el.removeEventListener('click', checker);
        if (el.textContent == div.ans) {
            el.classList.add('correct');
        }
    })
}

```

```

const div2 = document.createElement('div');
div2.classList.add('message');
div.append(div2);
if (sel.textContent === div.ans) {
    game.score++;
    div2.innerHTML = 'Correct';
    div2.style.color = 'green';
} else {
    div2.innerHTML = 'Wrong';
    div2.style.color = 'red';
}
contrs.style.display = 'block';
contrs.textContent = 'Next Question';
}

```

```

[
  {
    "question" : "What color is Grass",
    "ans" : "Green",
    "opts" : ["Blue","Red","Purple"]
  },
  {
    "question" : "What color is is Ocean",
    "ans" : "Blue",
    "opts" : ["Green","Red","Purple"]
  },
  {
    "question" : "What color is the sun",
    "ans" : "Yellow",

```

```

    "opts" : ["Blue","Red","Purple"]
  }
]

```

JavaScript Image Preview File Reader Example



Select images from your computer using a web input field, show a preview of selected images in your webpage. Make the images clickable to show a popup preview. Useful component to load images from local computer to a webpage and preview the images. Make the images selectable and popup as full size. Using JavaScript to handle the file contents and the output of the file contents to the page.

```

<!DOCTYPE html>
<html>
  <head><title>Image preview</title>

```

```

<style>
    .thumb{
        max-height:100px;
    }
    .popup{
        width:60%;
        height:40%;
        position:absolute;
        top:10%;
        left:20%;
        background-color:rgba(0,0,0,.66);
    }
    .popup img{
        max-width:100%;
        margin:auto;
        border:5px solid white;
        box-shadow: 10px 10px grey;
    }
</style>
</head>
<body>
    <input type="file" multiple accept="image/*"
id="myImages"/>
    <div class="preview"></div>
    <script src="image.js"></script>
</body>
</html>

```

```

const preview = document.querySelector('.preview');
const myInput = document.querySelector('#myImages');;

```

```
const popWindow = document.createElement('div');

const popImage = document.createElement('img');
popWindow.appendChild(popImage);
popWindow.style.display = 'none';
popWindow.classList.add('popup');
document.body.appendChild(popWindow);
popWindow.addEventListener('click', ()=>{
    popWindow.style.display = 'none';
})

myInput.addEventListener('change', upload);

function upload(e){
    const files = e.target.files;
    console.log(files);
    for(let i =0; i<files.length; i++){
        const file = files[i];
        const img = document.createElement('img');
        img.classList.add('thumb');
        img.file = file;
        img.addEventListener('click', (ee) =>{
            console.log(img.src);
            popup(img.src);
        })
        preview.appendChild(img);
        const reader = new FileReader();
        reader.onload = ((myImg) =>{
```



```

        return (ele)=>{
            myImg.src = ele.target.result;
        }
    })(img);
    reader.readAsDataURL(file);
}

function popup(img){
    popImage.setAttribute('src',img);
    popWindow.style.display = 'block';
}

```

JavaScript Interactive Dice Game with Page elements



Using CSS Grid and JavaScript create a visual dice game that allows the player to roll 2 dice and see who the winner is. Dynamically generate dice on the page, randomly roll the dice and

check which player wins. Use of JavaScript code to hide and show page elements, apply styling dynamically to page elements. JavaScript logic for game results and game play.

```
<!DOCTYPE html>
<html>

<head>
  <title>Dice Game</title>
  <style>
    * {
      box-sizing: border-box;
    }

    .viewer {
      display: inline-block;
      width: 220px;
      height: 180px;
      border: 1px solid black;
      padding: 5px;
    }

    .viewer span {
      display: block;
    }

    .viewer span:first-child {
      color: red;
      text-align: center;
      font-size: 1.5em;
    }
  </style>
</head>

<body>
  <div>
    <div class="viewer">
      <span>Player 1</span>
      <span>Player 2</span>
    </div>
  </div>
</body>
</html>
```

```
.viewer span:last-child {  
    color: black;  
    text-align: center;  
    font-size: 3em;  
}  
  
.output {  
    width: 440px;  
    text-align: center;  
    font-size: 2em;  
}  
  
.dot {  
    width: 100%;  
    height: 30px;  
    border-radius: 50%;  
    padding: 5px;  
}  
  
.parentDice {  
    border: 1px solid #ddd;  
    padding: 20px;  
    display: grid;  
    grid-gap: 1px;  
    grid-template-columns: repeat(3, 1fr);  
}
```

```

        .btn {
            width: 440px;
            height: 50px;
            background-color: red;
            color: white;
            font-size: 2em;
        }
    </style>
</head>

<body>
    <div class="gamearea">
        <div class="output"></div>
        <div class="viewer">
            <span class="playerName">Player 1</span>
            <span id="player1"></span>
        </div>
        <div class="viewer">
            <span class="playerName">Player 2</span>
            <span id="player2"></span>
        </div>
    </div>
    <button type="button" class="btn">Roll Dice</button>
    <script src="dice.js"></script>
</body>

</html>

```

```
const btn = document.querySelector('.btn');
```

```
const output = document.querySelector('.output');
const py1 = document.querySelector('#player1');
const py2 = document.querySelector('#player2');
const dice = [
  [5],
  [1, 9],
  [1, 5, 9],
  [1, 3, 7, 9],
  [1, 3, 5, 7, 9],
  [1, 3, 4, 6, 7, 9]
];

btn.addEventListener('click', () => {
  const rolls = [rollDice(6), rollDice(6)];
  let result;
  if (rolls[0] === rolls[1]) {
    result = "Draw";
  } else if (rolls[0] > rolls[1]) {
    result = "Player 1 Wins";
  } else {
    result = "Player 2 Wins";
  }
  updateDice(py1, rolls[0]);
  updateDice(py2, rolls[1]);
  //`${rolls[0]} vs ${rolls[1]} <br>
  output.innerHTML = `${result}`;
})
```

```
function updateDice(el, num) {
  el.innerHTML = "";
  const holder = builder(num);
  el.append(holder);
}

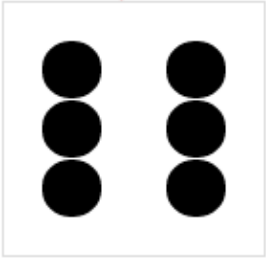
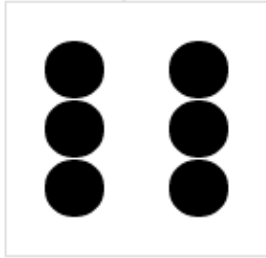
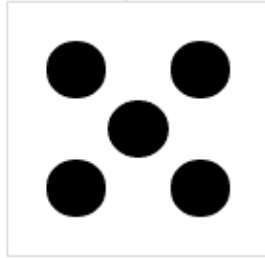
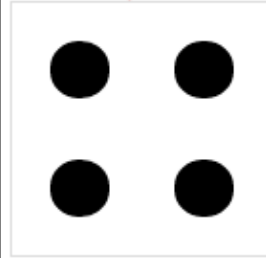
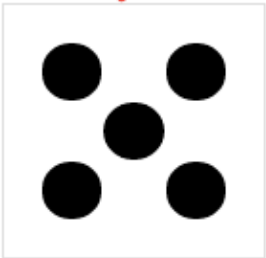
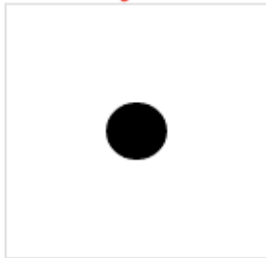
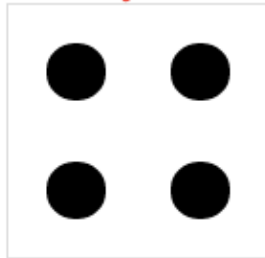
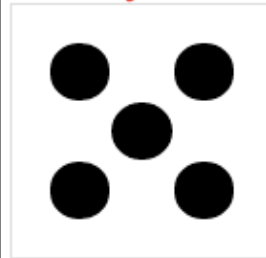
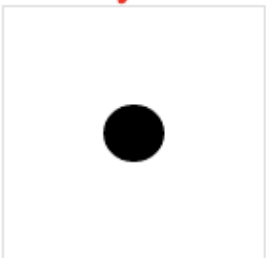
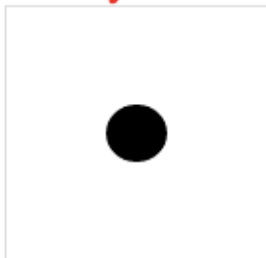
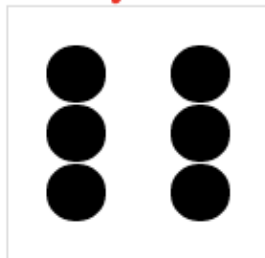
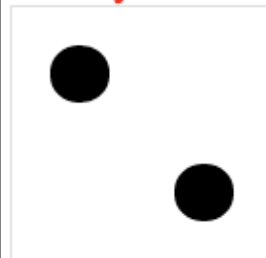
function builder(num) {
  const div = document.createElement('div');
  const addColor = dice[num - 1];
  console.log(addColor);
  for (let x = 1; x < 10; x++) {
    const el1 = document.createElement('div');
    el1.classList.add('dot');
    if (addColor.includes(x)) {
      el1.style.backgroundColor = "black";
    }
    div.append(el1);
  }
  div.classList.add('parentDice');
  return div;
}

function rollDice(num) {
  const rollValue = Math.floor(Math.random() * num) + 1;
  return rollValue;
}
```

JavaScript Dice Game Challenge Lesson

Highest Roll is 6

Winners List Player 1,Player 2,Player 11

Player 1 	Player 2 	Player 3 	Player 4 
Player 5 	Player 6 	Player 7 	Player 8 
Player 9 	Player 10 	Player 11 	Player 12 
RollDice			

Update the dice game to allow the user to select the number of players, and dynamically with JavaScript create all needed page elements for full game functionality. The game logic also needs to be updated to account for multiple winner probability and checking which of the rolls is the highest.

```
<!DOCTYPE html>
<html>
```

```
<head>

<title>Dice Game</title>

<style>

  * {

    box-sizing: border-box;

  }

  .viewer {

    display: inline-block;

    min-width:150px;

    height: 180px;

    border: 1px solid black;

    padding: 5px;

  }

  .viewer span {

    display: block;

  }

  .viewer span:first-child {

    color: red;

    text-align: center;

    font-size: 1.5em;

  }

  .viewer span:last-child {

    color: black;

    text-align: center;

  }


```



```
        font-size: 3em;
    }

    .output {
        width: 100%;
        text-align: center;
        font-size: 2em;
    }

    .eles{
        font-size:2em;
        display:block;
    }
    .dot {
        width: 100%;
        height: 30px;
        border-radius: 50%;
        padding: 5px;
    }
    .main{
        display:flex;
        justify-content :center;
    }
    .parentDice {
        border: 1px solid #ddd;
        padding: 20px;
        display: grid;
        grid-gap: 1px;
```

```
        grid-template-columns: repeat(3, 1fr);
    }

    .btn {
        width: 100%;
        height: 50px;
        background-color: red;
        color: white;
        font-size: 2em;
    }
</style>
</head>

<body>
    <div class="gamearea">
    </div>
    <script src="dice.js"></script>
</body>

</html>
```

```
const howMany = document.createElement('input');
const gameArea = document.querySelector('.gamearea');
howMany.setAttribute('type', 'number');
howMany.classList.add('eles');
howMany.value = 4;
const message = document.createElement('div');
gameArea.append(message);
message.classList.add('output');
```

```
const output = document.createElement('div');
gameArea.append(output);
output.classList.add('main');

output.append(howMany);
const btn1 = document.createElement('button');
btn1.textContent = "Start Game";
output.append(btn1);
btn1.addEventListener('click',startGame);
btn1.classList.add('eles');
const btn2 = document.createElement('button');
gameArea.append(btn2);
btn2.classList.add('btn');
btn2.textContent = "RollDice";
btn2.style.display = 'none';
const game = {players:[]};
const dice = [
    [5],
    [1, 9],
    [1, 5, 9],
    [1, 3, 7, 9],
    [1, 3, 5, 7, 9],
    [1, 3, 4, 6, 7, 9]
];

function startGame(){
    //console.log(howMany.value);
```

```

output.innerHTML = "";
btn2.style.display = 'block';
for(let x=0;x<howMany.value;x++){
    console.log(x);
    const temp = document.createElement('div');
    temp.classList.add('viewer');
    const span1 = document.createElement('span');
    span1.textContent = `Player ${x+1}`;
    temp.append(span1);
    const span2 = document.createElement('span');
    span2.textContent = "";
    temp.append(span2);
    output.append(temp);
    game.players.push(span2);
}
}

btn2.addEventListener('click', () => {
    const rolls = [];
    let highest = 0;
    const winners = [];
    for(let x=0;x<game.players.length;x++){
        let val = rollDice(6);
        rolls.push(val);
        if(val > highest){
            highest = val;

```

```

    }
    updateDice(game.players[x], val);
  }
  for(let x=0;x<rolls.length;x++){
    if(rolls[x] === highest){
      winners.push(`Player ${x+1}`);
    }
  }
  let winnerList = winners.toString();
  message.innerHTML = `Highest Roll is
${highest}<br><small>Winners List ${winnerList}</small>`;
})

function updateDice(el, num) {
  el.innerHTML = "";
  const holder = builder(num);
  el.append(holder);
}

function builder(num) {
  const div = document.createElement('div');
  const addColor = dice[num - 1];
  console.log(addColor);
  for (let x = 1; x < 10; x++) {
    const el1 = document.createElement('div');
    el1.classList.add('dot');
    if (addColor.includes(x)) {

```

```
        el1.style.backgroundColor = "black";
    }
    div.append(el1);
}
div.classList.add('parentDice');
return div;
}

function rollDice(num) {
    const rollValue = Math.floor(Math.random() * num) + 1;
    return rollValue;
}
```

JavaScript DOM Fun Projects Interactive DOM Elements

Explore how to make fun interactive web projects using JavaScript that allow you to select page elements and do amazing things with those elements. You can create interactive game, useful tools and components and a whole lot more.

JavaScript Tip Calculator Project

Total Bill =

Tip Percentage %

Tip should be \$15.00

Tip Calculator Project Part 1

Create a dynamic interface with JavaScript, add textNode and page elements with Code. Set Attributes of inputs to number type, add event listener to the button. Make a clickable button that gets values from the two input fields.

```
const output = document.querySelector('.output');

const div1 = createEle(output,'div','');
addText('Total Bill = ',div1);
const totalBill = createEle(div1,'input','');
const div2 = createEle(output,'div','');
addText('Tip Percentage % ',div2);
const totalPerc = createEle(div2,'input','');
totalBill.setAttribute('type','number');
totalPerc.setAttribute('type','number');
const btn = createEle(output,'button','Get total');
const total = createEle(output,'div','-');
```

```

btn.addEventListener('click',(e)=>{
    const tb = totalBill.value;
    const tp = totalPerc.value;
    console.log(tb,tp);
})

function addText(message,elParent){
    const temp = document.createTextNode(message);
    elParent.appendChild(temp);
}

function createEle(elParent,elType,html){
    const temp = document.createElement(elType);
    temp.innerHTML = html;
    return elParent.appendChild(temp);
}

```

Tip Calculator Project Part 2

Use the ternary operator to apply results of a condition in one statement line. Calculate the results of the input with conditions to ensure the input values are consistent with what we expect in the values. Apply to the number input type min and max attributes, also in the calculator do a final check, output the update check values to the input fields for the user.

condition ? exprIfTrue : exprIfFalse

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Conditional_Operator

```

const output = document.querySelector('.output');

const div1 = createEle(output,'div','');
addText('Total Bill = ',div1);

```



```

const totalBill = createEle(div1,'input','');
const div2 = createEle(output,'div','');
addText('Tip Percentage % ',div2);
const totalPerc = createEle(div2,'input','');
totalBill.setAttribute('type','number');
totalPerc.setAttribute('type','number');
totalBill.setAttribute('min','0');
totalPerc.setAttribute('max','100');
totalPerc.value = 15;
totalBill.value = 100;

const btn = createEle(output,'button','Get total');
const total = createEle(output,'div','-');

btn.addEventListener('click',(e)=>{
    let tb = totalBill.value;
    let tp = totalPerc.value;
    tp = (tp > 100) ? 100 : tp;
    tb = (tb < 0) ? 0 : tb;
    totalPerc.value = tp;
    totalBill.value = tb;
    console.log(tp);
    const tip = (tb*(tp/100)).toFixed(2);
    total.innerHTML = `Tip should be ${tip}`;
})

function addText(message,elParent){
    const temp = document.createTextNode(message);

```

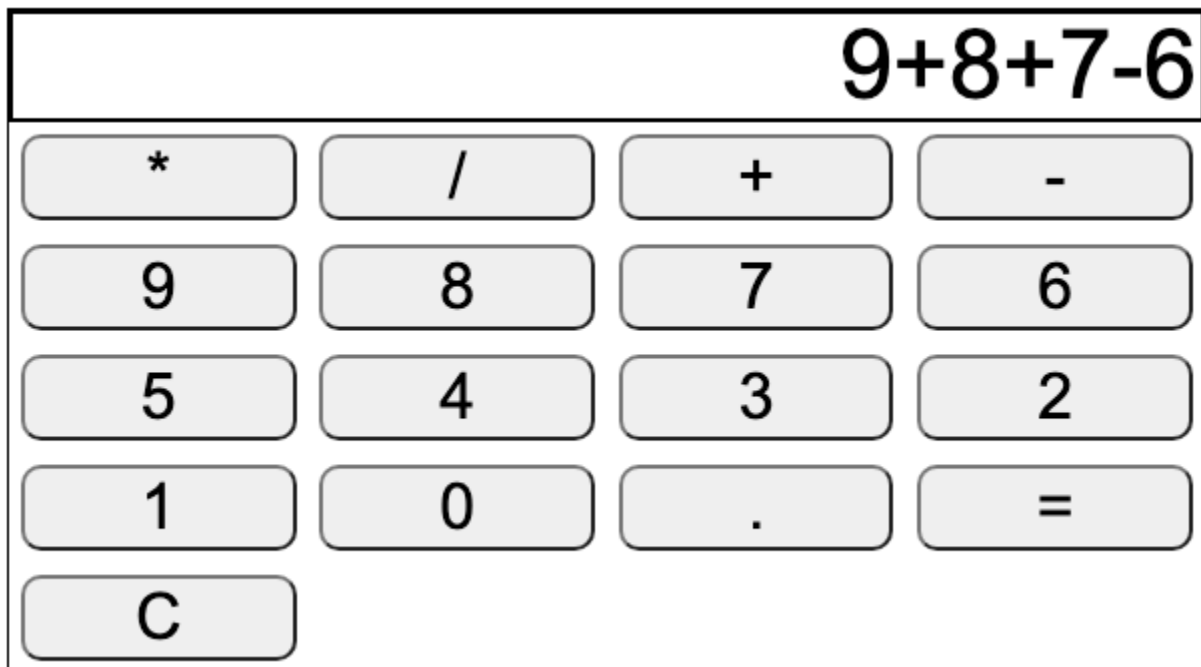
```

    elParent.appendChild(temp);
}

function createEle(elParent,elType,html){
    const temp = document.createElement(elType);
    temp.innerHTML = html;
    return elParent.appendChild(temp);
}

```

Pure JavaScript Calculator DOM page elements Project



JavaScript Calculator Part 1

In this lesson we create an interactive calculator with JavaScript. Press the keys on the calculator and have the values show in the display area. Keys are created from array data, elements and event listeners all added by JavaScript.

```
<!DOCTYPE html>
<html>
<head>
  <title>JavaScript Calculator</title>
  <style>
    *{
      box-sizing: border-box;
    }
    .main {
      width:100%;
      line-height: 50px;
      font-size: 3em;
      text-align: right;
    }
    .container {
      width: 100%;
    }
    .btn{
      width: 23%;
      font-size: 2em;
      margin:1%;
      border-radius: 10px;;
    }
  </style>
</head>
<body>
  <div class="output"></div>
  <script src="calculator.js"></script>
```

```
</body>
</html>
```

```
const output = document.querySelector('.output');
const keys =
['*', '/', '+', '-', '9', '8', '7', '6', '5', '4', '3', '2', '1', '0', '.', '=']
];
const spec = ['*', '/', '+', '-'];

output.style.maxWidth = '600px';
output.style.margin = 'auto';
output.style.border = '1px solid black';

const main = document.createElement('input');
main.setAttribute('type', 'text');
main.classList.add('main');
output.append(main);

const container = document.createElement('div');
container.classList.add('container');
output.append(container);

keys.forEach((key)=>{
  btnMaker(key, container);
})

function btnMaker(val, parentEl){
  const btn = document.createElement('button');
```

```

    btn.classList.add('btn');
    btn.textContent = val;
    btn.val = val;
    btn.addEventListener('click',(e)=>{
        addOutput(val);
    })
    parentEl.append(btn);
    return btn;
}

function addOutput(val){
    main.value += val;
}

```

JavaScript Calculator Part 2

Debugging of the JavaScript Calculator, how to use JavaScript functions `isNaN()` and `eval()` to calculate results. Conditions to ensure the display area can properly be calculated.

`isNaN(value)`

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/isNaN

`eval(string)`

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/eval

```

const output = document.querySelector('.output');
const keys =
['*', '/', '+', '-', '9', '8', '7', '6', '5', '4', '3', '2', '1', '0', '.', '=',
'C'];
const spec = ['*', '/', '+', '-'];
let dec = false;
output.style.maxWidth = '600px';

```

```
output.style.margin = 'auto';
output.style.border = '1px solid black';

const main = document.createElement('input');
main.setAttribute('type', 'text');
main.classList.add('main');
output.append(main);

const container = document.createElement('div');
container.classList.add('container');
output.append(container);

keys.forEach((key)=>{
  btnMaker(key, container);
})

function btnMaker(val, parentEl){
  const btn = document.createElement('button');
  btn.classList.add('btn');
  btn.textContent = val;
  btn.val = val;
  btn.addEventListener('click', (e)=>{
    addOutput(val);
  })
  parentEl.append(btn);
  return btn;
}
```

```
function addOutput(val){
    //console.log(val);
    let temp = main.value;
    let boo = true;
    let lastOne = temp.charAt(temp.length-1);
    //console.log(isNaN(lastOne));
    //console.log(isNaN(val));
    if(val === 'C'){
        main.value = '';
        boo = false;
    }else if(val === '='){
        evalOutput(lastOne);
        boo = false;
    }else if( val === '.'){
        if(dec){
            boo = false;
        }else{
            dec = true;
        }
    }
    if(isNaN(val) && lastOne.length === 0){
        boo = false;
    }
    if(isNaN(lastOne) && isNaN(val) ){
        boo = false;
    }
    if(boo){
        main.value += val;
    }
}
```

```

    }

}

function evalOutput(lastOne){
    if(!isNaN(lastOne)){
        main.value = eval(main.value);
    }
}

```

JavaScript Calculator Part 3

Add conditions to fix any issues with the calculator. The display area content will be evaluated by JavaScript so you need to ensure that the formula in the display area does not have any errors. Also ensure you can only have one decimal place and that you can clear the decimal place on clear. `Object.prototype.toString()` , `String.prototype.includes()` , `parseInt()`, `Number()`

Use of `.toString()` method to convert object to string

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Object/toString

Use of `.includes()` method to check if decimal is already in the display area

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/String/includes

Use of `parseInt()` vs `Number()` how you can return available digits in a string. Get the display area content ensure infinity does not show in the display area.

```

const output = document.querySelector('.output');
const keys = ['*', '/', '+', '-', '9', '8', '7', '6', '5', '4',
'3', '2', '1', '0', '.', '=', 'C'];
const spec = ['*', '/', '+', '-'];
let dec = false;

```



```
output.style.maxWidth = '600px';
output.style.margin = 'auto';
output.style.border = '1px solid black';

const main = document.createElement('input');
main.setAttribute('type', 'text');
main.classList.add('main');
output.append(main);

const container = document.createElement('div');
container.classList.add('container');
output.append(container);

keys.forEach((key) => {
  btnMaker(key, container);
})

function btnMaker(val, parentEl) {
  const btn = document.createElement('button');
  btn.classList.add('btn');
  btn.textContent = val;
  btn.val = val;
  btn.addEventListener('click', (e) => {
    addOutput(val);
  })
  parentEl.append(btn);
  return btn;
}
```

```

function togBorder(bc){
    main.style.borderColor = bc;
}

function addOutput(val) {
    //console.log(val);
    let temp = main.value;
    let boo = true;
    let lastOne = temp.charAt(temp.length - 1);
    //console.log(isNaN(lastOne));
    //console.log(isNaN(val));
    togBorder('black');

    if (isNaN(val) && lastOne.length === 0) {
        boo = false;
        togBorder('red');
    }
    if (isNaN(lastOne) && isNaN(val)) {
        boo = false;
        togBorder('red');
    }
    if (val === 'C') {
        main.value = '';
        dec = false;
        togBorder('black');
        boo = false;
    } else if (val === '=') {

```

```

        evalOutput(lastOne);
        boo = false;
    } else if (val === '.') {
        if (dec) {
            boo = false;
            togBorder('red');
        } else {
            dec = true;
        }
    }
    if (boo) {
        main.value += val;
    }
}

function evalOutput(lastOne) {
    if (!isNaN(lastOne)) {
        let temp = eval(main.value);
        temp = !parseInt(temp)? 0 : temp;
        temp = temp.toString();
        dec = temp.includes('.');
        console.log(dec);
        main.value = temp;
    }
}

```

```

<!DOCTYPE html>
<html>

```

```
<head>
  <title>JavaScript Calculator</title>
  <style>
    * {
      box-sizing: border-box;
    }

    .main {
      width: 100%;
      line-height: 50px;
      font-size: 3em;
      text-align: right;
      border: 2px solid black;
    }

    .container {
      width: 100%;
    }

    .btn {
      width: 23%;
      font-size: 2em;
      margin: 1%;
      border-radius: 10px;
      ;
    }
  </style>
```


<https://developer.mozilla.org/en-US/docs/Web/API/Window/cancelAnimationFrame> - Cancel Animation Frame

<https://developer.mozilla.org/en-US/docs/Web/API/window/requestAnimationFrame> Animation Frame

```
<!DOCTYPE html>
<html>
<head>
  <title>Drop Game</title>
  <style>
    .game {
      width:90%;
      margin:auto;
      border: 1px solid black;
      text-align:center;
    }
    .scoreBoard{
      background-color:rgb(15, 15, 15);
      color:white;
      padding:10px;
    }
    .gameBoard{
      background-color:paleturquoise;
      height:500px;
      position:relative;
      overflow:hidden;
    }
    .message{
      background-color:darkkhaki;
      padding:10px;
```

```
        font-family: Verdana, Geneva, Tahoma, sans-serif;
    }

    .btn{
        display: block;
        width: 100%;
        height: 50px;
        padding: 10px;
        border-radius: 20px;
        font-size: 1.5em;
    }

    .bubble, .baddy{
        width: 30px;
        height: 30px;

        border-radius: 50%;
        line-height: 30px;
        font-size: 1.2em;
        color: white;
        position: absolute;
        left: 50%;
        top: 100px;
        cursor: crosshair;
    }
</style>
</head>
<body>
    <div class="game"></div>
    <script src="drops.js"></script>
```

```
</body>
```

```
</html>
```

```
const container = document.querySelector('.game');
const scoreBoard = maker(container, 'div', 'scoreBoard', 'SCORE');
const gameBoard =
maker(container, 'div', 'gameBoard', 'GAMEBOARD');
const message = maker(container, 'div', 'message', 'MESSAGE');
const items =
["&#8509;", "&#9730;", "&#9731;", "&#9728;", "&#9732;", "&#9733;", "&#
9743;", "&#9762;"];
const game = {score:0, ani:{}, total:0, counter:0, ready:0, bad:0};
const btn = maker(container, 'button', 'btn', 'Click to Start');
btn.addEventListener('click', startGame);

function startGame(){
  btn.style.display = 'none';
  game.score = 0;
  game.total = 50;
  game.ready = 0;
  game.counter = 0;
  game.bad = 10;
  gameBoard.innerHTML = "";
  message.innerHTML = "Click the Bubbles";
  scoreUpdater();
  game.ani = requestAnimationFrame(mover);
}
```



```

function badBubbles(){
  const bubble = maker(container,'div','baddy',"&#9760;");
  const cSize = gameBoard.getBoundingClientRect();
  gameBoard.append(bubble);
  bubble.speed = ran(2,10);
  bubble.style.backgroundColor = 'red';
  bubble.style.transform = `scale(${ran(0.5,3)})`;
  bubble.style.left = ran(0,(cSize.width-30)) + 'px';
  bubble.style.top = ran(0,500) + 500 + 'px';
  bubble.addEventListener('mouseover',(e)=>{
    game.score--;
    gameBoard.style.backgroundColor = 'red';
  })
  bubble.addEventListener('mouseout',(e)=>{
    game.score--;
    gameBoard.style.backgroundColor = '';
  })
}

function genBubbles(){
  scoreUpdater();
  items.sort(()=>Math.random() - .5);
  const letter = items[0];
  const bubble = maker(container,'div','bubble',letter);
  const cSize = gameBoard.getBoundingClientRect();
  gameBoard.append(bubble);
  bubble.speed = ran(2,10);

```

```

        bubble.dir = ran(0,10) -5;
        bubble.style.backgroundColor = `rgba(${ran(0,255)},
${ran(0,255)}, ${ran(0,255)}, 0.5)`;
        bubble.style.transform = `scale(${ran(0.5,3)})`;
        bubble.style.left = ran(0,(cSize.width-30)) + 'px';
        bubble.style.top = ran(0,500) + 500 + 'px';
        bubble.addEventListener('mouseover',(e)=>{
            game.score+=10;
            game.counter++;
            scoreUpdater();
            bubble.remove();
            //check end game
            if((game.ready-game.counter)==0){
                message.innerHTML = 'Game Over';
                cancelAnimationFrame(game.ani);
                btn.style.display = 'block';
            }
        })
    }

function mover(){
    if(game.bad>0){
        badBubbles();
        game.bad--;
    }
    if(game.ready < game.total){
        console.log(game);
        game.ready++;
    }
}

```

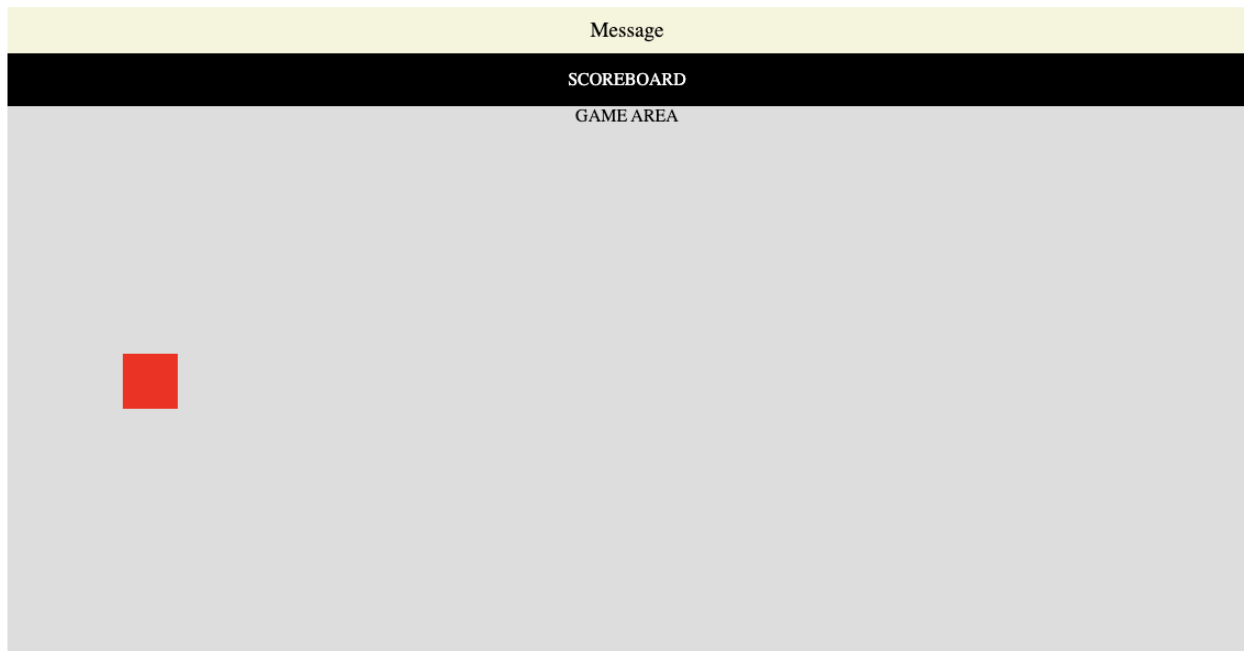
```

    genBubbles();
}
const allBaddy = document.querySelectorAll('.baddy');
allBaddy.forEach((bubble)=>{
    const pos = [bubble.offsetLeft,bubble.offsetTop];
    const speed = bubble.speed;
    pos[1]-=speed;
    if(pos[1]<-100){
        bubble.remove();
        badBubbles();
    }
    bubble.style.top = pos[1] + 'px';
    bubble.style.left = pos[0] + 'px';
})
const allBubbles = document.querySelectorAll('.bubble');
allBubbles.forEach((bubble)=>{
    const pos = [bubble.offsetLeft,bubble.offsetTop];
    const speed = bubble.speed;
    pos[1]-=speed;
    if(pos[1]<-100){
        bubble.remove();
        game.score--;
        genBubbles();
        scoreUpdater();
    }
    bubble.style.top = pos[1] + 'px';
    bubble.style.left = pos[0] + 'px';
    //console.log(pos);

```

```
    })  
    game.ani = requestAnimationFrame(mover);  
}  
  
function scoreUpdater(){  
    scoreBoard.innerHTML = `Your Score : ${game.score}`;  
    message.innerHTML = `Bubbles Left :  
${game.ready-game.counter}`;  
}  
  
function ran(min,max){  
    return Math.floor(Math.random()*(max-min)+min);  
}  
  
function maker(parent,eleType,myClass,html){  
    const ele = document.createElement(eleType);  
    ele.classList.add(myClass);  
    ele.innerHTML = html;  
    parent.append(ele);  
    return ele;  
}
```

How to move a Page Element With JavaScript DOM Mover Example



Element DOM Mover - how to move an element on the page within a parent element. Keypress tracking and movement with request animation frame. Select a page element and update the style properties to move an element on the page using JavaScript. Use of event listeners like key presses, tracking of key presses and applying logic to make action within the web page.

```
<!DOCTYPE html>
<html>
<head>
  <title>Game Basics</title>
  <style>
    *{
      box-sizing:border-box;
    }
    .game{
      width:90%;
      margin:auto;
      text-align:center;
```

```
}  
  .message{  
    padding:10px;  
    font-size:1.2em;  
    background-color:beige;  
  }  
  .scoreboard{  
    background-color:black;  
    color:white;  
    padding:15px;  
  }  
  .gamePlay{  
    height:500px;  
    width:100%;  
    background-color:#ddd;  
    position:relative;  
    overflow:hidden;  
  }  
  .box{  
    width:50px;  
    height:50px;  
    background-color:red;  
    position:absolute;  
    z-index:100;  
  }  
</style>  
</head>  
<body>
```

```

<div class="game"></div>
<script src="game.js"></script>
</body>
</html>

```

```

const gameArea = document.querySelector('.game');
const message = maker(gameArea, 'div', 'message', 'Message');
const scoreBoard =
maker(gameArea, 'div', 'scoreboard', 'SCOREBOARD');
const gamePlay = maker(gameArea, 'div', 'gamePlay', 'GAME AREA');
const box = maker(gamePlay, 'div', 'box', ' ');
const game = {ani:null,x:0,y:0,speed:5};
const keyz = {
  ArrowDown: false,
  ArrowLeft: false,
  ArrowRight: false,
  ArrowUp: false
};

window.addEventListener('keydown', (e)=>{
  if(e.code in keyz){keyz[e.code] = true;}
})
window.addEventListener('keyup', (e)=>{
  if(e.code in keyz){keyz[e.code] = false;}
})
game.ani = window.requestAnimationFrame(mover);

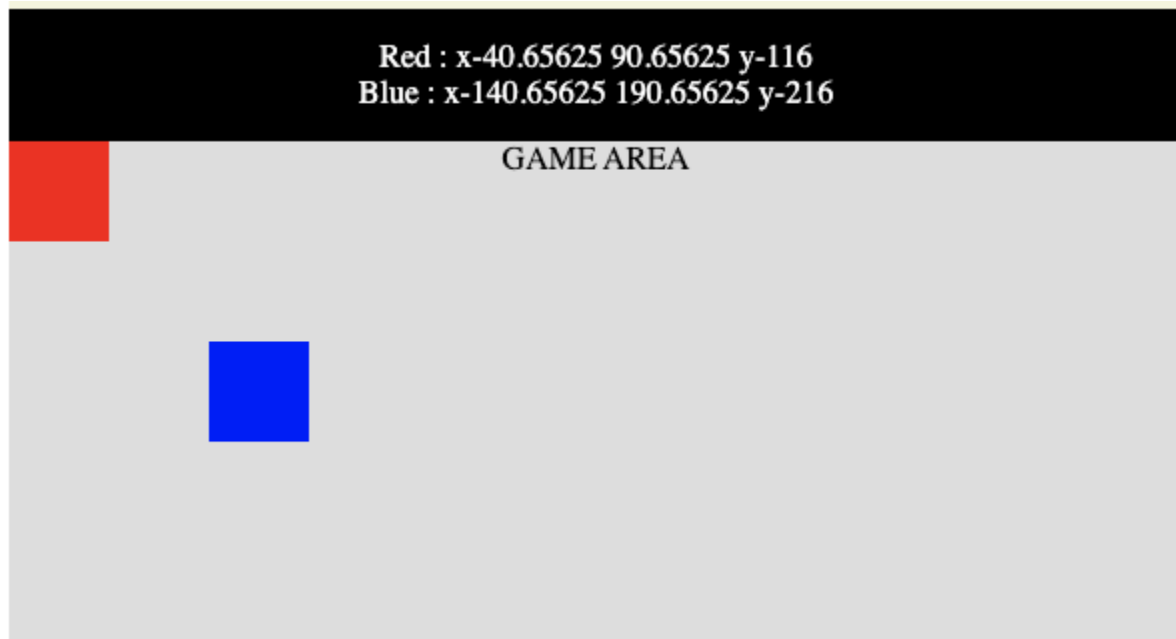
function mover(){
  if(keyz.ArrowDown){

```

```
        game.y+=game.speed;
    }else if(keyz.ArrowUp){
        game.y-=game.speed;
    }
    if(keyz.ArrowLeft){
        game.x-=game.speed;
    }else if(keyz.ArrowRight){
        game.x+=game.speed;
    }
    box.style.left = game.x + 'px';
    box.style.top = game.y + 'px';
    game.ani = window.requestAnimationFrame(mover);
}

function maker(parent,myType,myClass,html){
    const ele = document.createElement(myType);
    parent.append(ele);
    ele.classList.add(myClass);
    ele.innerHTML = html;
    return ele;
}
```


Collision Detection between Page elements with JavaScript DOM



Code to check intersection of the coordinate between 2 elements on the page. Collision detection is an important part of any game to check if there is overlap of the position between two separate elements on the page. This lesson will demo how to check for the overlap and return a response.

```
<!DOCTYPE html>
<html>
<head>
  <title>Game Basics</title>
  <style>
    *{
      box-sizing:border-box;
    }
    .game{
      width:90%;
      margin:auto;
      text-align:center;
    }
  </style>
</head>
<body>
```

```
.message{
    padding:10px;
    font-size:1.2em;
    background-color:beige;
}

.scoreboard{
    background-color:black;
    color:white;
    padding:15px;
}

.gamePlay{
    height:500px;
    width:100%;
    background-color:#ddd;
    position:relative;
    overflow:hidden;
}

.box{
    width:50px;
    height:50px;
    background-color:red;
    position:absolute;
    z-index:100;
}

</style>
</head>
<body>
    <div class="game"></div>
```

```

    <script src="game.js"></script>
  </body>
</html>

```

```

const gameArea = document.querySelector('.game');
const message = maker(gameArea, 'div', 'message', 'Message');
const scoreBoard =
maker(gameArea, 'div', 'scoreboard', 'SCOREBOARD');
const gamePlay = maker(gameArea, 'div', 'gamePlay', 'GAME AREA');
const box = maker(gamePlay, 'div', 'box', ' ');
const box1 = maker(gamePlay, 'div', 'box', ' ');
box1.style.backgroundColor = 'blue';
const game = {ani:null,x:0,y:0,speed:5};
const item1 = {x:100,y:100};
const keyz = {
  ArrowDown: false,
  ArrowLeft: false,
  ArrowRight: false,
  ArrowUp: false
};

window.addEventListener('keydown', (e)=>{
  if(e.code in keyz){keyz[e.code] = true;}
})
window.addEventListener('keyup', (e)=>{
  if(e.code in keyz){keyz[e.code] = false;}
})
game.ani = window.requestAnimationFrame(mover);

```

```

function mover(){
    if(keyz.ArrowDown){
        game.y+=game.speed;
    }else if(keyz.ArrowUp){
        game.y-=game.speed;
    }
    if(keyz.ArrowLeft){
        game.x-=game.speed;
    }else if(keyz.ArrowRight){
        game.x+=game.speed;
    }
    box.style.left = game.x + 'px';
    box.style.top = game.y + 'px';
    box1.style.left = item1.x + 'px';
    box1.style.top = item1.y + 'px';
    if(checkCol(box,box1)){
        message.textContent = 'Hit!!';
        setInterval(()=>{message.textContent = ' '},3000);
        item1.x =
Math.floor(Math.random()*(gamePlay.offsetWidth-50));
        item1.y =
Math.floor(Math.random()*(gamePlay.offsetHeight-50));
    };
    game.ani = window.requestAnimationFrame(mover);
}

function checkCol(ele1,ele2){
    const a = ele1.getBoundingClientRect();

```

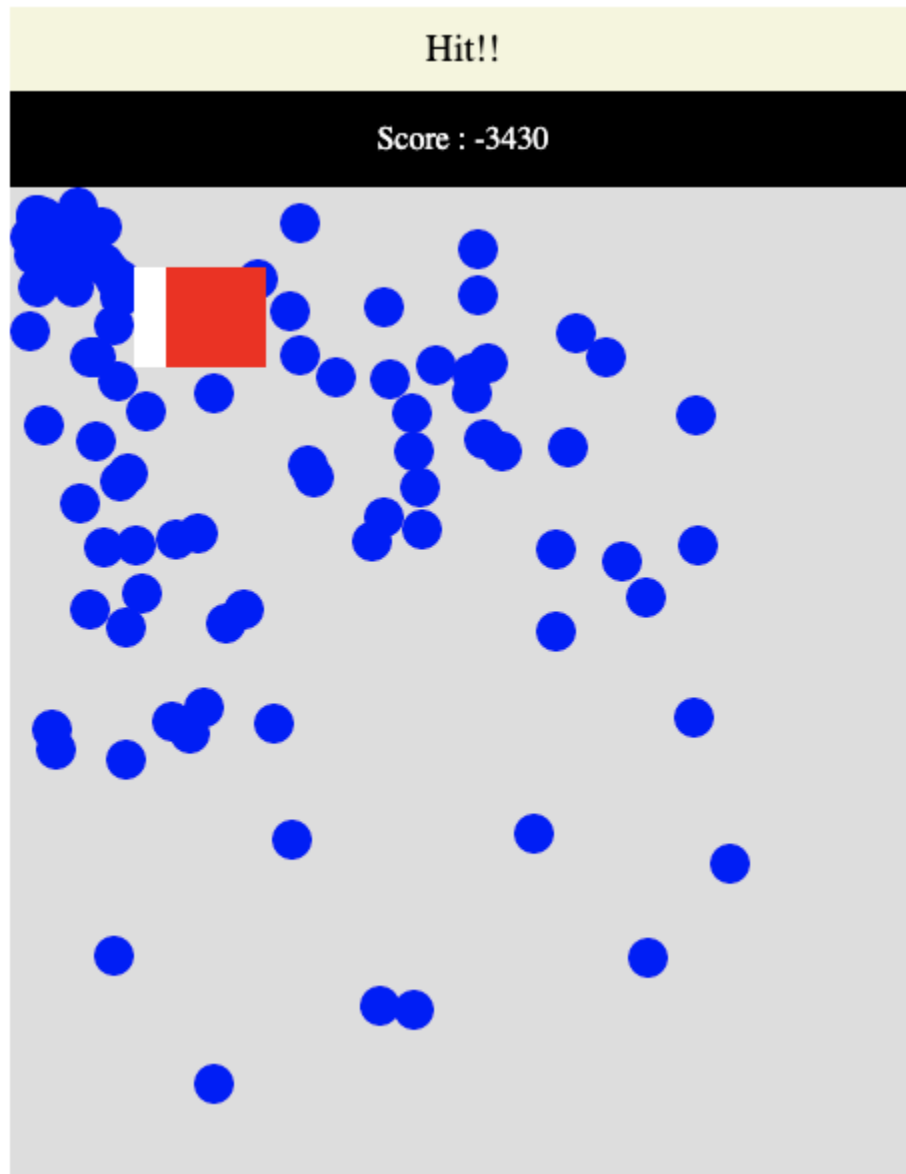
```

    const b = ele2.getBoundingClientRect();
    scoreBoard.innerHTML = `Red : x-${a.left} ${a.right}
y-${a.top}<br>`;
    scoreBoard.innerHTML += `Blue : x-${b.left} ${b.right}
y-${b.top}`;
    let hit = !((a.right < b.left) || (a.left > b.right) || (a.bottom
< b.top) || (a.top > b.bottom));
    if(hit){
        scoreBoard.style.backgroundColor = 'red';
    }else{
        scoreBoard.style.backgroundColor = 'black';
    }
    return hit;
}

function maker(parent,myType,myClass,html){
    const ele = document.createElement(myType);
    parent.append(ele);
    ele.classList.add(myClass);
    ele.innerHTML = html;
    return ele;
}

```

JavaScript DOM Interactive Game



Create a simple catch and avoid game with JavaScript and DOM page element manipulation. Move page elements by keyboard events of arrow keys. Use of animation frames to create smooth page transitions. Collision detection of page elements to check for overlap and create gameplay.

```
<!DOCTYPE html>
<html>
<head>
  <title>Game Basics</title>
```

```
<style>
  *{
    box-sizing:border-box;
  }
  .game{
    width:90%;
    margin:auto;
    text-align:center;
  }
  .message{
    padding:10px;
    font-size:1.2em;
    background-color:beige;
  }
  .scoreboard{
    background-color:black;
    color:white;
    padding:15px;
  }
  .gamePlay{
    height:500px;
    width:100%;
    background-color:#ddd;
    position:relative;
    overflow:hidden;
  }
  .box{
    width:50px;
```

```

        height:50px;
        background-color:white;
        position:absolute;
        z-index:100;
    }
    .box1{
        width:20px;
        height:20px;
        background-color:blue;
        position:absolute;
        z-index:50;
        border-radius:50%;
    }
</style>
</head>
<body>
    <div class="game"></div>
    <script src="game.js"></script>
</body>
</html>

```

```

const gameArea = document.querySelector('.game');
const message = maker(gameArea,'div','message','Message');
const scoreBoard =
maker(gameArea,'div','scoreboard','SCOREBOARD');
const gamePlay = maker(gameArea,'div','gamePlay','');
const box = maker(gamePlay,'div','box',' ');

```



```

const game =
{ani:null,x:gamePlay.offsetWidth/2-25,y:gamePlay.offsetHeight/2,
speed:5,enemies:100,score:0};
const enemies = [];
const box1 = maker(gamePlay,'div','box',' ');
box1.style.backgroundColor = 'red';
box1.pos = {x:0,y:0,dx:0,dy:0,dir:10,speed:5};
const keyz = {
  ArrowDown: false,
  ArrowLeft: false,
  ArrowRight: false,
  ArrowUp: false
};

window.addEventListener('keydown',(e)=>{
  if(e.code in keyz){keyz[e.code] = true;}
})
window.addEventListener('keyup',(e)=>{
  if(e.code in keyz){keyz[e.code] = false;}
})
game.ani = window.requestAnimationFrame(mover);

function mover(){
  //Gen Enemies
  box1.pos.dir--;
  if(box1.pos.dir < 0){
    box1.pos.speed = ran(0,6);
  }
}

```

```

        box1.pos.dx = (box1.pos.x < game.x) ? box1.pos.speed :
box1.pos.speed *-1;
        box1.pos.dy = (box1.pos.y < game.y) ? box1.pos.speed
:box1.pos.speed *-1;
        box1.pos.dir = ran(10,20);
    }
    box1.pos.x+=box1.pos.dx;
    box1.pos.y+=box1.pos.dy;
    box1.style.left = box1.pos.x + 'px';
    box1.style.top = box1.pos.y + 'px';
    if(checkCol(box,box1)){
        message.textContent = 'They Got YOU';
        game.score--;
        scoreBoard.innerHTML = `Score : ${game.score}`;
    }

    if(enemies.length < game.enemies ){
        if(ran(0,5) == 0 ){
            const obj = {
                x:ran(0,gamePlay.offsetWidth-20),
                y:ran(0,gamePlay.offsetHeight-20),
                dx:0,
                dy:5,
                dir:10,
                ele: maker(gamePlay,'div','box1',' ')
            };
            obj.ele.style.backgroundColor = 'blue';
            enemies.push(obj);

```

```

    }

}

if(keyz.ArrowDown && game.y < gamePlay.offsetHeight-50){
    game.y+=game.speed;
}else if(keyz.ArrowUp && game.y> 0){
    game.y-=game.speed;
}

if(keyz.ArrowLeft){
    game.x-=game.speed;
}else if(keyz.ArrowRight){
    game.x+=game.speed;
}

box.style.left = game.x + 'px';
box.style.top = game.y + 'px';

enemies.forEach((ene)=>{
    ene.dir--;
    if(ene.dir<0){
        ene.dir = ran(10,50);
        ene.dy = ran(0,10)-5;
        ene.dx = ran(0,10)-5;
    }
    if(ene.y < 0){ene.dy=5;}
    if(ene.x <0){ene.dx=5;}
    if(ene.y > gamePlay.offsetHeight-50){ene.dy=-5;}
    if(ene.x > gamePlay.offsetWidth-50){ene.dx=-5;}
    ene.y+=ene.dy;
    ene.x+=ene.dx;

```

```

        ene.ele.style.left = ene.x + 'px';
        ene.ele.style.top = ene.y + 'px';
        if(checkCol(box,ene.ele)){
            message.textContent = 'Hit!!';
            ene.y = 0;
            ene.x = 0;
            game.score++;
            scoreBoard.innerHTML = `Score : ${game.score}`;
        }

    })
    game.ani = window.requestAnimationFrame(mover);
}

function ran(min,max){
    return Math.floor(Math.random()*(max-min)+min);
}

function checkCol(ele1,ele2){
    const a = ele1.getBoundingClientRect();
    const b = ele2.getBoundingClientRect();
    let hit = !((a.right < b.left)|| (a.left > b.right)|| (a.bottom
< b.top)|| (a.top > b.bottom));
    if(hit){
        scoreBoard.style.backgroundColor = 'red';
    }else{

```

```
        scoreBoard.style.backgroundColor = 'black';
    }
    return hit;
}

function maker(parent,myType,myClass,html){
    const ele = document.createElement(myType);
    parent.append(ele);
    ele.classList.add(myClass);
    ele.innerHTML = html;
    return ele;
}
```

Course resource Guide
Laurence Svekis
basescripts.com