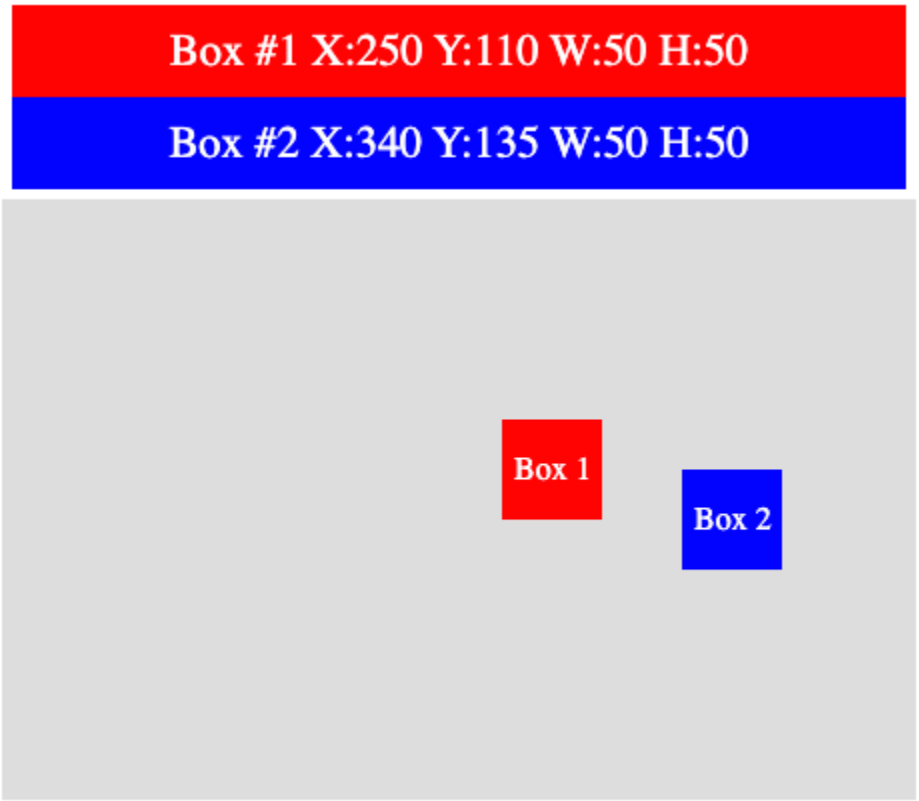
JavaScript DOM element Collision detection

The overlap of elements is what can be referred to as collision detection. This is a standard part of most games, when moving page elements to be able to detect the position and calculate if they are overlapping.

This exercise is designed to test and learn more about how to check for overlap of elements that move on the page. It will demonstrate how to set up movement of page elements, and output the values needed to calculate the collision between two page elements.



Set up movement of page elements and track the position exercise :

1. Create a function that allows for the creation of page elements as we can then dynamically add elements to the page as needed. Give the function a name of adder() to get the parent element that the new element will be appended to. Create the element using the string value of the element tag. Add the inner HTML content for the element and also add a class using the string value.

```
function adder(parent,t,html,c){
   const ele = document.createElement(t);
   ele.innerHTML = html;
   ele.classList.add(c);
   return parent.appendChild(ele);
}
```

2. Create 4 page elements, 2 boxes that will be moved, and 2 areas to output the box coordinates and dimensions.
3. Create a global game object that can track the animation frame, contains the x,y (horizontal,vertical) position of each box, and that contains the h,w (height and width) of each box. These values then can be used to update the position.
4. Set up an object named keyz to track which keyboard keys will be used for element movement. Add 8 key code names as property names and set the values to false. This will be updated once the key is pressed.

```
const keyz = {
   ArrowLeft:false, ArrowRight:false, ArrowUp:false,
ArrowDown:false,
   KeyA:false, KeyW:false, KeyS:false, KeyZ:false,
};
```

5. Add event listeners on the document that listen for keydown and keyup events. If the keys from the keyz object pressed match the object property name, set the value to true on press down and false on up. This can now be used for the element movement.
6. Create and Launch the mover() function which will handle page element movement on key presses.
7. Within the mover() function add conditions to check which keys are true, update the game object x,y for each box accordingly. Also apply a condition to check to ensure that before the movement is allowed

that the element is within the boundaries of the main container object element.

8. Update the style values of properties left and top for each box with the new game x,y values for each.
9. Output the x,y,h,w of each element to the page as they move; these values should change with the new x,y coordinates of the element. These will be used to calculate the collision of the elements.

```
<!DOCTYPE html>
<html>

<head>
   <title>JavaScript Course</title>
   <style>
       .container {
           position: relative;
           background-color: #ddd;
           width: 90vw;
           margin: auto;

       }

       .box {
           position: absolute;
           width: 50px;
           height: 50px;
           left: 0;
           top: 0px;
           line-height: 50px;
           text-align: center;
```

```
            color: white;
        }

        .info {
            background-color: white;
            padding: 5px;
            text-align: center;
            font-size: 1.4em;
            color: white;
        }

        .info div {
            padding: 10px;
        }

        .game {
            position: relative;
            min-height: 300px;
        }
    </style>
</head>

<body>
    <div class="container"></div>
    <script src="app8.js"></script>
</body>

</html>
```

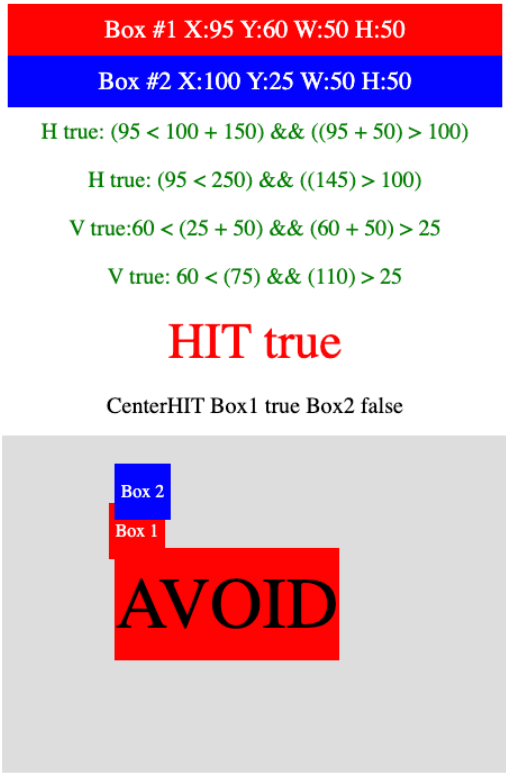
```
const main = document.querySelector('.container');
const message = adder(main, 'div', '', 'info');
const output1 = adder(message, 'div', 'Content 1', 'output');
const output2 = adder(message, 'div', 'Content 2', 'output');

const gameBoard = adder(main, 'div', '', 'game');
const box1 = adder(gameBoard, 'div', 'Box 1', 'box');
box1.style.backgroundColor = 'red';
output1.style.backgroundColor = 'red';
const box2 = adder(gameBoard, 'div', 'Box 2', 'box');
box2.style.backgroundColor = 'blue';
output2.style.backgroundColor = 'blue';
box2.style.left = '100px';
const game = {
   move: {},
   x1: box1.offsetLeft,
   y1: box1.offsetTop,
   x2: box2.offsetLeft,
   y2: box2.offsetTop,
   speed: 5,
   w1: box1.offsetWidth,
   h1: box1.offsetHeight,
   w2: box2.offsetWidth,
   h2: box2.offsetHeight
};
const keyz = {
   ArrowLeft: false,
   ArrowRight: false,
```

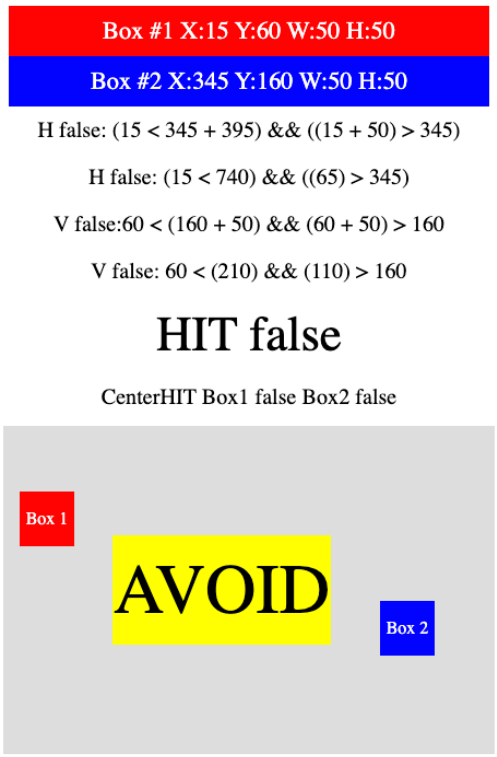
```
    ArrowUp: false,
    ArrowDown: false,
    KeyA: false,
    KeyW: false,
    KeyS: false,
    KeyZ: false,
};

document.addEventListener('keydown', (e) => {
    if (e.code in keyz) {
        keyz[e.code] = true;
    }
})
document.addEventListener('keyup', (e) => {
    if (e.code in keyz) {
        keyz[e.code] = false;
    }
})

window.addEventListener('DOMContentLoaded', mover);

function mover() {
    const dim = [gameBoard.offsetWidth - box1.offsetWidth,
gameBoard.offsetHeight - box1.offsetHeight];
    //Box 1
    if (keyz.ArrowRight && game.x1 < dim[0]) game.x1 +=
game.speed;
    if (keyz.ArrowLeft && game.x1 > 0) game.x1 -= game.speed;
```

```
    if (keyz.ArrowDown && game.y1 < dim[1]) game.y1 +=
game.speed;
    if (keyz.ArrowUp && game.y1 > 0) game.y1 -= game.speed;
    box1.style.left = `${game.x1}px`;
    box1.style.top = `${game.y1}px`;

    //Box 2
    if (keyz.KeyS && game.x2 < dim[0]) game.x2 += game.speed;
    if (keyz.KeyA && game.x2 > 0) game.x2 -= game.speed;
    if (keyz.KeyZ && game.y2 < dim[1]) game.y2 += game.speed;
    if (keyz.KeyW && game.y2 > 0) game.y2 -= game.speed;
    box2.style.left = `${game.x2}px`;
    box2.style.top = `${game.y2}px`;

    output1.innerHTML = `Box #1 X:${game.x1} Y:${game.y1}
W:${game.w1} H:${game.h1}`;
    output2.innerHTML = `Box #2 X:${game.x2} Y:${game.y2}
W:${game.w2} H:${game.h2}`;
    game.move = window.requestAnimationFrame(mover);
}

function adder(parent, t, html, c) {
    const ele = document.createElement(t);
    ele.innerHTML = html;
    ele.classList.add(c);
    return parent.appendChild(ele);
}
```

Collision detection Checker of DOM page elements Part 2

Source Code Collision detection with JavaScript DOM elements example



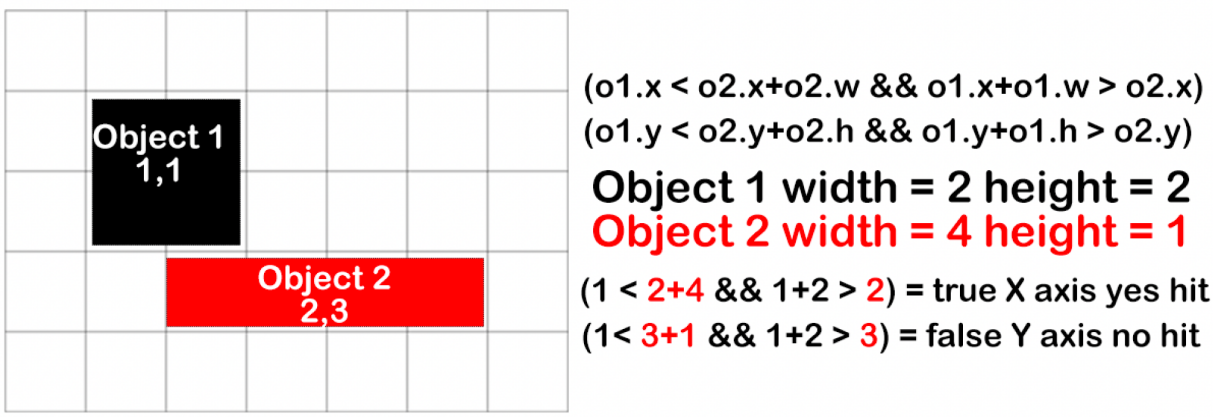


To check if any two elements are overlapping you need both element, x,y height and width values. These can then be used to calculate the corners to see overlap on the horizontal axis and overlap on the vertical axis. If both horizontal and vertical are overlapping then there is a collision occurring between the two elements on the page.

Using getBoundingClientRect() will return the dimension of an element. This method returns an object of values which properties include, bottom, height, left, right, top, width, x, y

```
box1.getBoundingClientRect()
▼DOMRect {x: 98.15625, y: 457, width: 50, height: 50, top: 457, …}
    bottom: 507
    height: 50
    left: 98.15625
    right: 148.15625
    top: 457
    width: 50
    x: 98.15625
    y: 457
  ▶ [[Prototype]]: DOMRect
```

These values for the element can then be used in the collision formula to check for overlap.



X-Horizontal check formula :

```
(a.x < (b.x + b.width)) && ((a.x + a.width) > b.x);
```

y-Vertical check formula :

```
(a.y < (b.y+b.height)) && ((a.y + a.height) > b.y);
```

Moving exercise to check for element overlap:

1. create move output areas to show the current stats of the element and the collision values. Update the style of the element if an overlap occurs.

2. Create a new element in the center to use within the collision check function.
3. Create a new function that requires two parameters, one for each element to check. Get the boundaries of the element.
4. Create a function col() which can check the boundaries and return a boolean value if the two elements are overlapping. Use this formula to check for overlap of the elements between the various boxes on the screen.
5. Update element styling and output info styling depending on the result of the collision detection.

```
<!DOCTYPE html>
<html>

<head>
   <title>JavaScript Course</title>
   <style>
       .container {
           position: relative;
           background-color: #ddd;
           width: 90vw;
           margin: auto;

       }

       .box {
           position: absolute;
           width: 50px;
           height: 50px;
           left: 0;
           top: 0px;
```

```
    line-height: 50px;
    text-align: center;
    color: white;
}

.info {
    background-color: white;
    padding: 5px;
    text-align: center;
    font-size: 1.4em;
    color: white;
}
.output1{
    color:black;
    font-size: 0.9em;
    padding:0px;
}
.blocker{
    width:200px;
    left:100px;
    top:100px;
    position:absolute;
    height:100px;
    background-color:yellow;
    text-align:center;
    line-height:100px;
    font-size:4em;
}
```

```
        .output2{
            color:black;
            font-size:2em;
        }
        .info div {
            padding: 10px;
        }

        .game {
            position: relative;
            min-height: 300px;
        }
    </style>
</head>

<body>
    <div class="container"></div>
    <script src="app8.js"></script>
</body>

</html>

const main = document.querySelector('.container');
const message = adder(main, 'div', '', 'info');
const output1 = adder(message, 'div', 'Content 1', 'output');
const output2 = adder(message, 'div', 'Content 2', 'output');
const output3 = adder(message, 'div', 'Content 3', 'output1');
const output4 = adder(message, 'div', 'Content 4', 'output1');
const output5 = adder(message, 'div', 'Content 5', 'output1');
```

```
const output6 = adder(message, 'div', 'Content 6', 'output1');
const output7 = adder(message, 'div', 'Content 7', 'output2');
const output8 = adder(message, 'div', 'Content 8', 'output1');
const gameBoard = adder(main, 'div', '', 'game');
const box3 = adder(gameBoard,'div','AVOID','blocker');
const box1 = adder(gameBoard, 'div', 'Box 1', 'box');
box1.style.backgroundColor = 'red';
output1.style.backgroundColor = 'red';
const box2 = adder(gameBoard, 'div', 'Box 2', 'box');
box2.style.backgroundColor = 'blue';
output2.style.backgroundColor = 'blue';
box2.style.left = '100px';

const game = {
   move: {},
   x1: box1.offsetLeft,
   y1: box1.offsetTop,
   x2: box2.offsetLeft,
   y2: box2.offsetTop,
   speed: 5,
   w1: box1.offsetWidth,
   h1: box1.offsetHeight,
   w2: box2.offsetWidth,
   h2: box2.offsetHeight
};
const keyz = {
   ArrowLeft: false,
   ArrowRight: false,
```

```
    ArrowUp: false,
    ArrowDown: false,
    KeyA: false,
    KeyW: false,
    KeyS: false,
    KeyZ: false,
};

document.addEventListener('keydown', (e) => {
    if (e.code in keyz) {
        keyz[e.code] = true;
    }
})
document.addEventListener('keyup', (e) => {
    if (e.code in keyz) {
        keyz[e.code] = false;
    }
})

window.addEventListener('DOMContentLoaded', mover);

function mover() {
    const dim = [gameBoard.offsetWidth - box1.offsetWidth,
gameBoard.offsetHeight - box1.offsetHeight];
    //Box 1
    if (keyz.ArrowRight && game.x1 < dim[0]) game.x1 +=
game.speed;
    if (keyz.ArrowLeft && game.x1 > 0) game.x1 -= game.speed;
```

```
    if (keyz.ArrowDown && game.y1 < dim[1]) game.y1 +=
game.speed;
    if (keyz.ArrowUp && game.y1 > 0) game.y1 -= game.speed;
    box1.style.left = `${game.x1}px`;
    box1.style.top = `${game.y1}px`;

    //Box 2
    if (keyz.KeyS && game.x2 < dim[0]) game.x2 += game.speed;
    if (keyz.KeyA && game.x2 > 0) game.x2 -= game.speed;
    if (keyz.KeyZ && game.y2 < dim[1]) game.y2 += game.speed;
    if (keyz.KeyW && game.y2 > 0) game.y2 -= game.speed;
    box2.style.left = `${game.x2}px`;
    box2.style.top = `${game.y2}px`;

    output1.innerHTML = `Box #1 X:${game.x1} Y:${game.y1}
W:${game.w1} H:${game.h1}`;
    output2.innerHTML = `Box #2 X:${game.x2} Y:${game.y2}
W:${game.w2} H:${game.h2}`;
    const o3 = game.x1 < (game.x2 + game.w2) && (game.x1+
game.w1) > game.x2;
    const o4 = game.y1 < (game.y2 + game.h2) && (game.y1+
game.h1) > game.y2;
    output3.style.color = (o3) ? 'green' : 'black';
    output4.style.color = (o3) ? 'green' : 'black';
    output5.style.color = (o4) ? 'green' : 'black';
    output6.style.color = (o4) ? 'green' : 'black';
```

```
   output3.innerHTML = `H ${o3}: (${game.x1} < ${game.x2} +
${game.x2 + game.w2}) && ((${game.x1} + ${game.w1}) >
${game.x2})`;
   output4.innerHTML = `H ${o3}: (${game.x1} < ${game.x2 +
game.x2 + game.w2}) && ((${game.x1+ game.w1}) > ${game.x2})`;
   output5.innerHTML = `V ${o4}:${game.y1} < (${game.y2} +
${game.h2}) && (${game.y1} + ${game.h1}) > ${game.y2}`;
   output6.innerHTML = `V ${o4}: ${game.y1} < (${game.y2 +
game.h2}) && (${game.y1 + game.h1}) > ${game.y2}`;
   const val1 = col(box1,box2);
   output7.style.color = val1 ? 'red' : 'black';
   output7.innerHTML = `HIT ${val1}`;
   const val2 = col(box1,box3);
   const val3 = col(box2,box3);
   box3.style.backgroundColor = val2 ? 'red' : '';
   if(!val2){
       box3.style.backgroundColor = val3 ? 'blue' : '';
   }
   output8.innerHTML = `CenterHIT Box1 ${val2} Box2 ${val3}`;
   game.move = window.requestAnimationFrame(mover);
}

function col(el1,el2){
   const a = el1.getBoundingClientRect();
   const b = el2.getBoundingClientRect();
   //console.log(a);
   //const tempX = (a.x < (b.x + b.width)) && ((a.x + a.width) >
b.x);
```

```
   //const tempY = (a.y < (b.y+b.height)) && ((a.y + a.height) >
b.y);
   return (a.x < (b.x + b.width)) && ((a.x + a.width) > b.x) &&
(a.y < (b.y+b.height)) && ((a.y + a.height) > b.y);
}

function adder(parent, t, html, c) {
   const ele = document.createElement(t);
   ele.innerHTML = html;
   ele.classList.add(c);
   return parent.appendChild(ele);
}
```