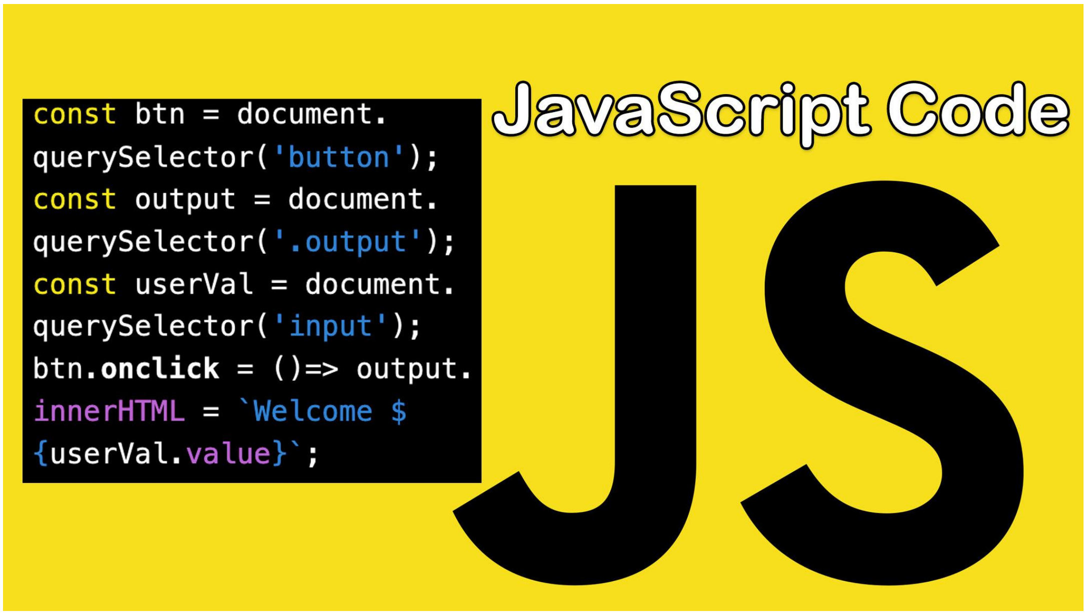
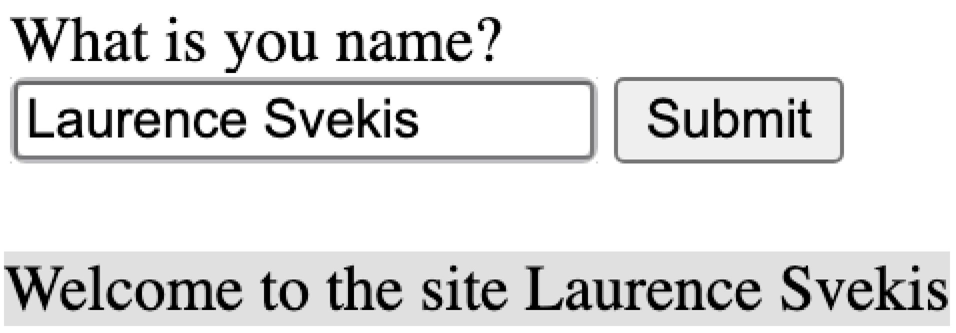
JavaScript DOM coding exercises



```
const btn = document.
querySelector('button');
const output = document.
querySelector('.output');
const userVal = document.
querySelector('input');
btn.onclick = ()=> output.
innerHTML = `Welcome $
{userVal.value}`;
```

Do you want to learn more about JavaScript and how elements work? They are objects which can hold values just like any other objects in JavaScript. The three exercises below will demonstrate creating page elements with JavaScript, how to add values to the element objects, and how to update page elements. There is also an exercise on how they work with the page element objects, and the difference between function expressions and function declarations.

Web page dynamic welcome message



Welcome to the site Laurence Svekis

Create a page welcome message :

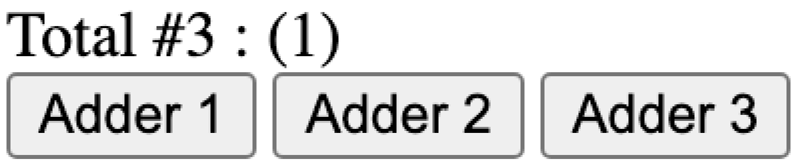
1. Get the user name in the input field
2. When the button is pressed add an event that get the user name and creates a welcome message on the page
3. Add a check to ensure the length of the input is larger than 3 characters long

```html
<!DOCTYPE html>
<html>
<head>
   <title>JavaScript Course</title>
</head>
<body>
<div class="output">What is your name?</div>
<input type="text" name="userName">
<button>Submit</button>
<script src="app.js"></script>
</body>
</html>
```

```
const output = document.querySelector('.output');
const userVal =
document.querySelector('input[name="userName"]');
const btn = document.querySelector('button');
userVal.style.borderColor = '#333';
btn.onclick = ()=>{
   if(userVal.value.length > 3){
       const message = `Welcome to the site ${userVal.value}`;
       output.textContent =  message;
       userVal.style.display = 'none';
       btn.style.display = 'none';
   }
   else{
       output.textContent = 'Name length must be 3+ characters';
       userVal.style.borderColor = 'red';
   }
}
```

Function Expression vs Function Declaration

This Counter


Total #3 : (1)

Create 3 page counters using different functions, expression, declaration and arrow format. Counters will all work independently as the value of the total is contained within the instance of the function object using this.

Page counters with functions :

1. Select all the page buttons
2. Create a function expression that will increment the counter on button press
3. Create a function expression with the arrow format that will track and increment the count on the page
4. Create a function declaration that will track and increment the count on the page.

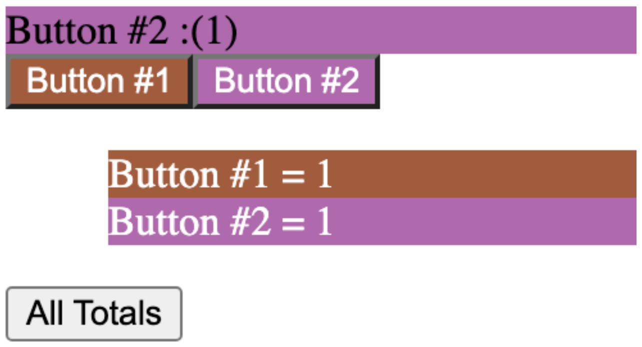
```
<!DOCTYPE html>
<html>
<head>
   <title>JavaScript Course</title>
</head>
<body>
<div class="output"></div>
<button class="btn1">Adder 1</button>
<button class="btn2">Adder 2</button>
<button class="btn3">Adder 3</button>
```

```
<script src="app1.js"></script>
</body>
</html>
```

```
const btn1 = document.querySelector('.btn1');
const btn2 = document.querySelector('.btn2');
const btn3 = document.querySelector('.btn3');
const output = document.querySelector('.output');
//Function Expression
const adder2 =  function(){
   if(!this.total) {
       this.total = 1;
   }else{
       this.total++;
   }
   output.textContent = `Total #2 : (${this.total})`;
}
const adder3 = ()=>{
   if(!this.total) {
       this.total = 1;
   }else{
       this.total++;
   }
   output.textContent = `Total #3 : (${this.total})`;
}
btn1.onclick = adder1;
btn2.onclick = adder2;
```

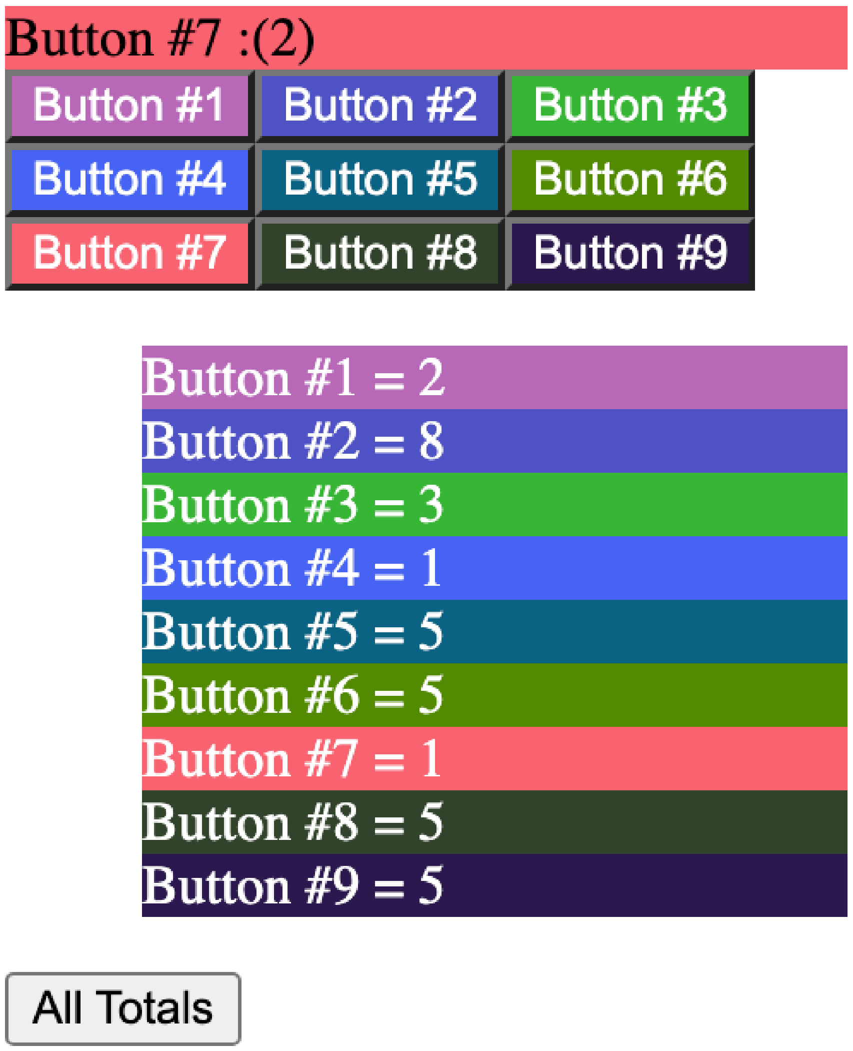
```
btn3.onclick = adder3;

///Function Declaration

function adder1(){
   if(!this.total) {
       this.total = 1;
   }else{
       this.total++;
   }
   output.textContent = `Total #1 : (${this.total})`;
}
```

Dynamic JavaScript DOM page counters Element Objects examples



Button #2 :(1)

Button #1 = 1
Button #2 = 1



Button #7 :(2)

Button #1 = 2
Button #2 = 8
Button #3 = 3
Button #4 = 1
Button #5 = 5
Button #6 = 5
Button #7 = 1
Button #8 = 5
Button #9 = 5

Dynamically create page buttons that can be used to count totals separately. Create a button to output all the result totals. Only using JavaScript NO HTML elements

Dynamic Page counters with JavaScript :

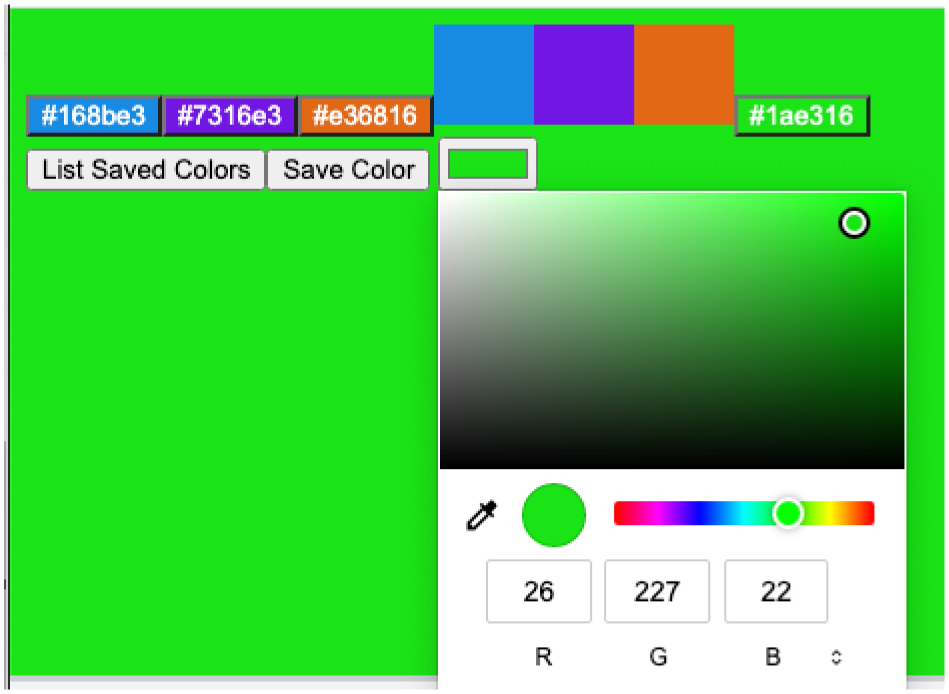
1. Create a global object to set the number of buttons to be created
2. Create a main page element that will contain all the new elements
3. Create a function to create page elements, adding the element type to a parent object. Include a parameter in the function for the inner content of the element.
4. Loop through and create all the buttons, set a default total of 0 for each one. On click create a function that will update and output the current value of this element total. You can add styling to the buttons
5. Add a class of “btn” to each new button so that they can be distinguished from the main total tally button.   Create a button that will output all the current button total results.
6. When the tally button is clicked create a function that will select all the elements with a class of “btn” and loop through them.
7. For each element with the class of “btn” create a new list item element, output that into an unordered list on the main page. The list item contents should show the element total and the element textContent. You can also select the style to match the button style property values.

```html
<!DOCTYPE html>
<html>
<head>
   <title>JavaScript Course</title>
</head>
<body>
<script src="app2.js"></script>
</body>
</html>
```

```
const val = {
   num : 2
};
const main =  elemaker('div',document.body,'');
const output =  elemaker('div',main,'');
for(let i=0;i<val.num;i++){
   const btn =  elemaker('button',main,`Button #${i+1}`);
   btn.total = 0;
   btn.style.backgroundColor =
'#'+Math.random().toString(16).slice(2,8);
   btn.style.color = 'white';
   btn.classList.add('btn');
   btn.onclick = adder;
}
const output1 =  elemaker('div',main,'');
const btnMain = elemaker('button',main,`All Totals`);
btnMain.onclick = ()=>{
   const btns = document.querySelectorAll('.btn');
   output1.innerHTML = '';
   const ul = elemaker('ul',output1,'');
   btns.forEach((ele,index)=>{
       const li = elemaker('li',ul,`${ele.textContent} =
${ele.total}`);
       li.style.backgroundColor = ele.style.backgroundColor;
       li.style.color  = ele.style.color;
   })
}
```

```
function elemaker(eleT,parent,html){
   const ele =  document.createElement(eleT);
   ele.innerHTML = html;
   return parent.appendChild(ele);
}

function adder(e){
   this.total++;
   output.style.backgroundColor= this.style.backgroundColor;
   output.innerHTML = `${this.textContent} :(${this.total})`;
}
```

DOM create Page elements adding style



Create interactive elements that can store the current color value input an array, for later use. Also creates buttons to update the body background color to the value of the button text. Color style and events with Dynamic Elements DOM.

Page elements Events and Style Exercise :

1. Using JavaScript, create two buttons on the page.
2. Add a click event to the first button, that when pressed should list out all the stored color values from the holding array. When pressed create page elements that have the color value from the holding array, and are clickable elements. Once clicked the element should be removed from the page.
3. The second button should store the color value from the input field into a holding array.
4. The second button should also create a page element button, that has the text content of the input field color value.
5. On change of the input value update the background color to match the input field value.

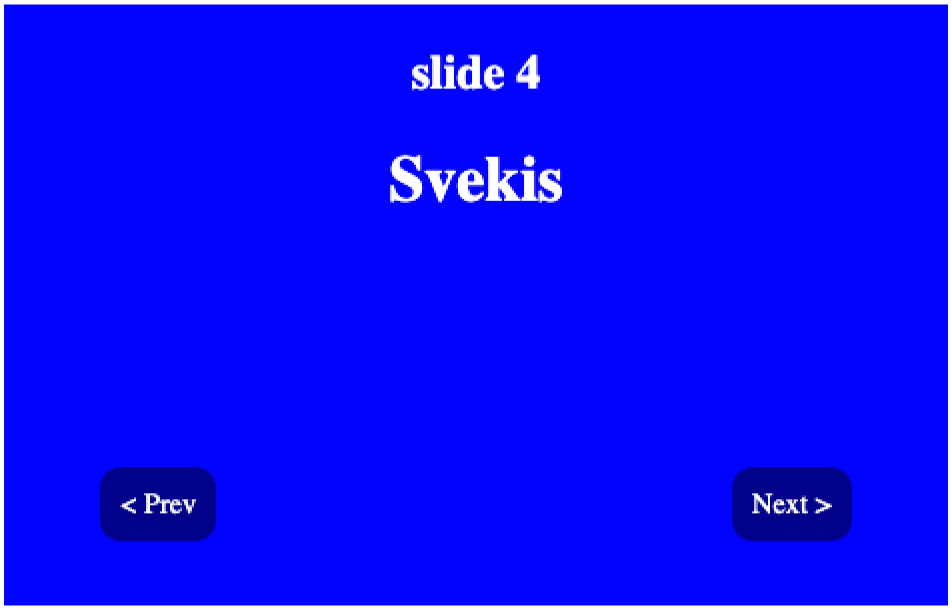
```
<!DOCTYPE html>
```

```
<html>
<head>
   <title>JavaScript Course</title>
</head>
<body>
   <input type="color">
   <script src="app4.js"></script>
</body>
</html>
```

```
const myInput = document.querySelector('input[type="color"]');
console.log(myInput);
const holder = [];
const main = document.createElement('div');

const btn = document.createElement('button');
const btn2 = document.createElement('button');
btn.textContent = "Save Color";
btn2.textContent = "List Saved Colors";
document.body.prepend(btn);
document.body.prepend(btn2);
document.body.prepend(main);
console.log(main);

btn2.onclick = ()=>{
   holder.forEach((ele)=>{
       const span = document.createElement('span');
       main.append(span);
```

```
        span.style.width = '50px';
        span.style.height = '50px';
        span.style.display = 'inline-block';
        span.style.backgroundColor = ele;
        span.onclick = ()=>{
            span.remove();}
    })
}


btn.onclick = ()=>{
    holder.push(myInput.value);
    //main.textContent = holder.toString();
    const btn1 = document.createElement('button');
    main.append(btn1);
    btn1.textContent = myInput.value;
    btn1.style.backgroundColor = myInput.value;
    btn1.style.color = 'white';
    btn1.onclick = ()=>{
        document.body.style.backgroundColor = btn1.textContent;
    }
}


myInput.onchange = (e)=>{
    console.log(e.target.value);
    console.log(myInput.value);
    document.body.style.backgroundColor = myInput.value;
    ///console.log(this.value);
}
```

Slider from Dynamic Elements using JSON data



Using JSON data, load the json file when the DOM content is loaded. Create the page slides dynamically with JavaScript code. Add interaction to navigate between slides listening for clicks on the buttons.

JSON slides navigate slides with JavaScript Exercise :

1. Select the main slider element as a variable named slider in JavaScript.
2. Create a function that gets invoked once the DOM content loads. Give the function a name of adder()
3. Within adder() create the slides, using the data from the JSON file apply the text title, html content, and styling to the slide elements. Add them to the page.

4. If it's the first item in the data then add the class of active to the first one only
5. Add an event listener to the buttons, when clicked check if it contains the next class, create a function named mover() to handle the result
6. Get the current element with the active class. Using traversing, get the next sibling to the current element, if no next element exists then select the first child of the parent slider element.
7. Get the previous element, if it's null then select the last child of the parent slider.
8. Remove the active class for the current element, add the active class to the new element either previous or next.

HTML and CSS

```
<!DOCTYPE html>
<html>

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

       .main {
           width: 90vw;
           margin: auto;
           background-color: black;
           position: relative;
           height: 300px;
```

```css
    overflow: hidden;
    color:white;
}

.slide {
    text-align:center;
    position: absolute;
    top: 0;
    left: 0;
    width: 100%;
    height: 100%;
    background-color: blue;
    opacity: 0;
}

.slide.active {
    opacity: 1;
}

.btn {
    position: absolute;
    bottom: 2rem;
    padding: 10px;
    background-color: rgba(0, 0, 0, 0.5);
    border-radius: 10px;
    font-size: 0.9em;
    color: white;
    cursor:pointer;
```

```
        }
        .btn:hover{
            background-color: rgb(0, 0, 0);
        }

        .next {
            right: 3rem;
        }

        .prev {
            left: 3rem;
        }
    </style>
</head>

<body>
    <div class="main">
        <div>
            <div class="slider">
            </div>
            <div class="prev btn">< Prev</div>
            <div class="next btn">Next ></div>
        </div>
    </div>
    <script src="app5.js"></script>
</body>

</html>
```

JAVASCRIPT CODE

```
const slider = document.querySelector('.slider');
const btns = document.querySelectorAll('.btn');
window.addEventListener('DOMContentLoaded',init);
btns.forEach(btn =>{
   btn.onclick = ()=>{
       console.log('clicked');
       mover(btn.classList.contains('next'));
   }
})

function mover(dir){
   const cur = document.querySelector('.active');
   cur.classList.remove('active');
   const nex = cur.nextElementSibling ||
slider.firstElementChild;
   const pre = cur.previousElementSibling ||
slider.lastElementChild;
   const newActive = (dir) ? nex : pre;
   newActive.classList.add('active');
}

function init(){
   //console.log('ready');
   fetch('app5.json')
       .then(response => response.json())
```

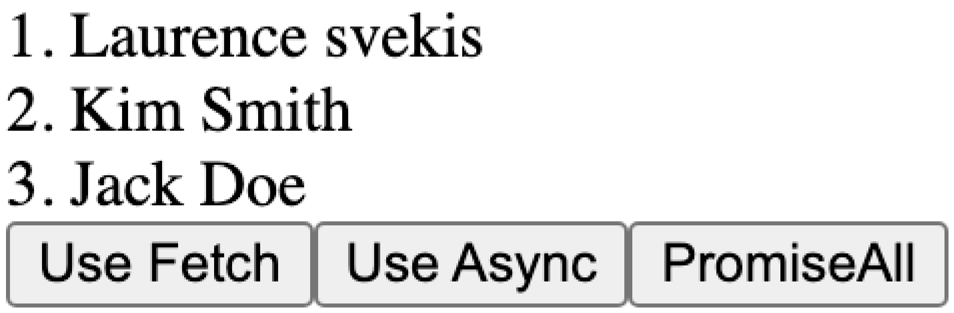
```
        .then(data =>{
            adder(data);
        })
}

function adder(data){
    //slider
    data.forEach((item,index)=>{
        const slide = document.createElement('div');
        slide.classList.add('slide');
        const title = document.createElement('h2');
        title.textContent = item.title;
        title.style.textAlign = 'center';
        if(index==0) slide.classList.add('active');
        const div  = document.createElement('div');
        div.innerHTML = item.html;
        slide.append(title);
        slide.append(div);
        slide.style.backgroundColor = item.back;
        slide.style.color = item.color;
        slider.append(slide);
    })
}
```

JSON file

```
[
    {
        "title": "slide 1",
        "html": "<h1>Laurence</h1>",
        "back": "purple",
        "color": "white"
    },
    {
        "title": "slide 2",
        "html": "<h1>Svekis</h1>",
        "back": "blue",
        "color": "white"
    },    {
        "title": "slide 4",
        "html": "<h1>Svekis</h1>",
        "back": "blue",
        "color": "white"
    },
    {
        "title": "slide 3",
        "html": "<h1>JavaScript</h1>",
        "back": "yellow",
        "color": "black"
    }
]
```

AJAX JavaScript Fetch with PromiseAll Method for multiple URLs

Exercise lesson to practice making fetch requests, using async and await with fetch and also how you can connect to multiple endpoints using promiseAll and return the results as one object value.


1. Laurence svekis
2. Kim Smith
3. Jack Doe

Fetch can be used to connect to external files, like JSON and return the results of those data files into JavaScript code. The data needs time to load, that is why promises are used to handle the data once it's ready and returned from the endpoint. Using async and await in the function can set up the promise to wait until a response is returned. Fetch can do the same with the chaining of then promises to handle the data once its arrived. You can also use promiseAll if there are multiple files that need to load data into one object. Once all the data is ready then the file contents can be used in the code and output on the page.

AJAX request example code exercise :

1. Create 3 page buttons to make requests. Set up an element on the page to show the output of the results.
2. Using the first button on click, connect to a json file and return the contents back using fetch. Output the results of the file onto the web page.
3. The second button uses async on the main request function, and await to move to the next function once the request completes. Output the results of the file data into the web page.
4. Create an array of the paths to several JSON files, with the same object structure in each file.

5. Once the third button is pressed, use Promise.all to make fetch requests to each JSON file. Nest the map function within the Promise.all request, so that the code can iterate through each JSON file. Once all the requests are completed then output the content into the web page from the JSON files.

HTML Code

```
<!DOCTYPE html>

<html>


<head>

   <title>JavaScript Course</title>

</head>


<body>

   <div class="main">

   </div>

   <script src="app6.js"></script>

</body>


</html>
```

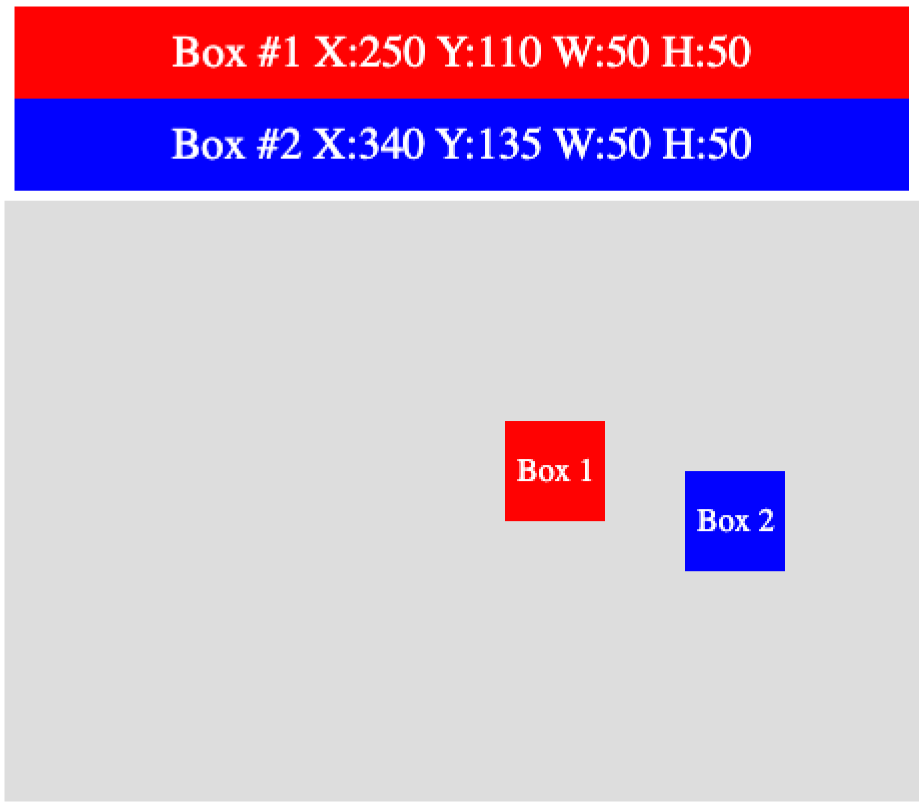
JavaScript Code

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

console.log(main);

const btn1 = document.createElement('button');

const btn2 = document.createElement('button');

const btn3 = document.createElement('button');

const output = document.createElement('div');

main.append(output);
main.append(btn1);
main.append(btn2);
main.append(btn3);
btn1.textContent = 'Use Fetch';
btn2.textContent = 'Use Async';
btn3.textContent = 'PromiseAll';
const urls = ['app6-1.json', 'app6-2.json', 'app6-3.json'];

btn1.onclick = fetchNow;
btn2.onclick = fetchData;
btn3.onclick = fetchAllUrls;

function fetchAllUrls() {
   Promise.all(urls.map(url =>
       fetch(url).then(res => res.json())
   )).then((data) => {
       let html = '';
       data.forEach((val, ind) => {
           console.log(val);
           html += `<div>${ind+1}. ${val.first}
${val.last}</div>`;
       })
       output.innerHTML = html;
   });
}



function fetchAllUrls2() {
   Promise.all([
       fetch(urls[0]).then(resp => resp.json()),
       fetch(urls[1]).then(resp => resp.json()),
       fetch(urls[2]).then(resp => resp.json()),
   ]).then(console.log);
}

function fetchNow() {
   fetch(urls[1])
       .then(response => response.json())
       .then(data => {
           console.log(data);
           output.textContent = `${data.first} ${data.last}`;
       })
       .catch(error => console.log(error));

}

async function fetchData() {
   let rep = await fetch(urls[0]);
   let data = await rep.json();
   console.log(data);
   output.textContent = `${data.first} ${data.last}`;
}

JSON #1
{
   "first" : "Laurence",
   "last" : "svekis"
}

JSON #2
{
   "first" : "Jack",
   "last" : "Doe"
}

JSON #3
{
   "first" : "Kim",
   "last" : "Smith"
}
```

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.


Box #1 X:250 Y:110 W:50 H:50

Box #2 X:340 Y:135 W:50 H:50

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.

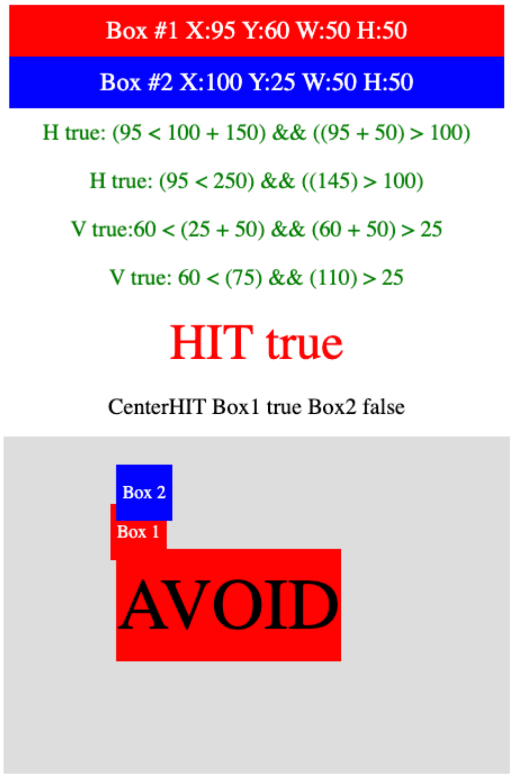
```
<!DOCTYPE html>
<html>

<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;
```

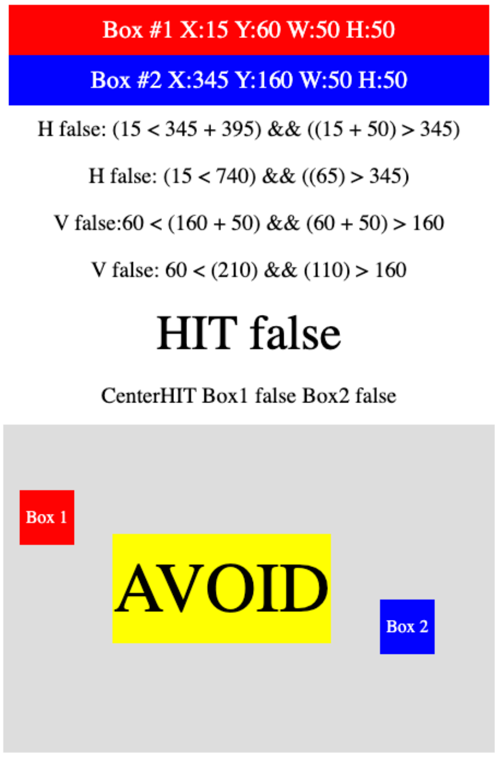
```
   if (keyz.ArrowUp && game.y1 > 0) game.y1 -= game.speed;
   box1.style.left = `${game.x1}px`;
   box1.style.top = `${game.y1}px`;

   //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



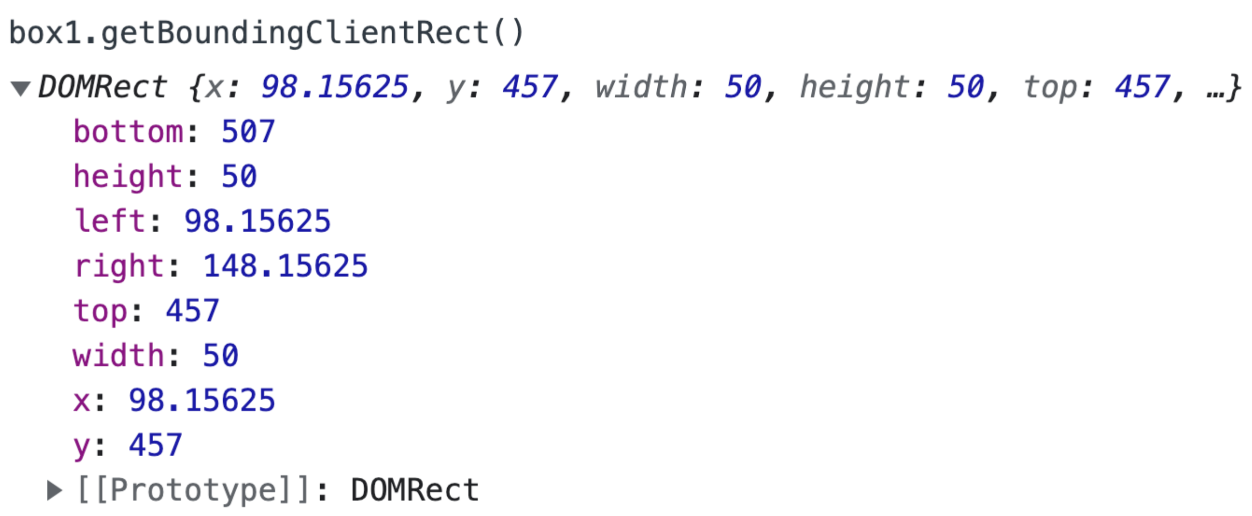
```
Box #1 X:95 Y:60 W:50 H:50
Box #2 X:100 Y:25 W:50 H:50
H true: (95 < 100 + 150) && ((95 + 50) > 100)
H true: (95 < 250) && ((145) > 100)
V true:60 < (25 + 50) && (60 + 50) > 25
V true: 60 < (75) && (110) > 25
HIT true
CenterHIT Box1 true Box2 false
```



```
Box #1 X:15 Y:60 W:50 H:50
Box #2 X:345 Y:160 W:50 H:50
H false: (15 < 345 + 395) && ((15 + 50) > 345)
H false: (15 < 740) && ((65) > 345)
V false:60 < (160 + 50) && (60 + 50) > 160
V false: 60 < (210) && (110) > 160
HIT false
CenterHIT Box1 false Box2 false
```

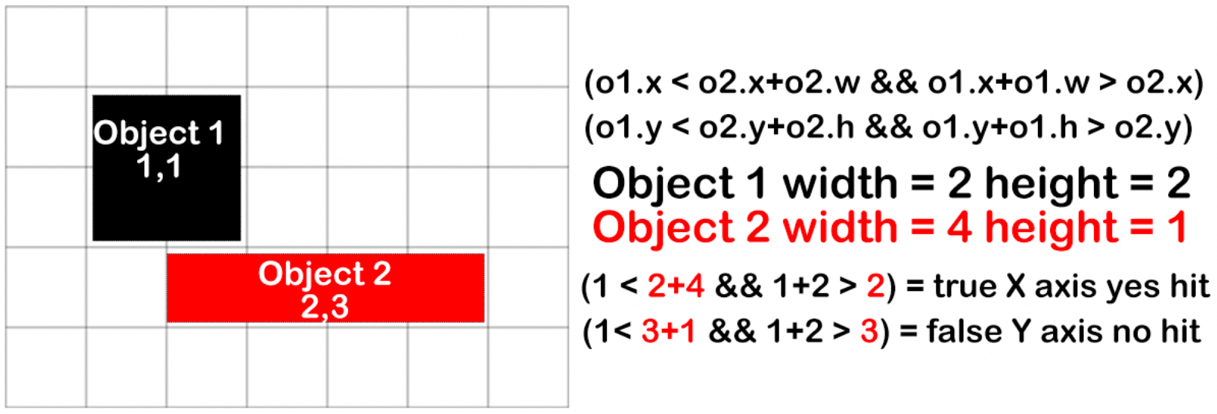
To check if any two elements are overlapping you need both element, x,y height and width values. These can then be used to calculate the corners to see overlap on the horizontal axis and overlap on the vertical axis. If both horizontal and vertical are overlapping then there is a collision occurring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



(o1.x < o2.x+o2.w && o1.x+o1.w > o2.x)
(o1.y < o2.y+o2.h && o1.y+o1.h > o2.y)

Object 1 width = 2 height = 2
Object 2 width = 4 height = 1

(1 < 2+4 && 1+2 > 2) = true X axis yes hit
(1< 3+1 && 1+2 > 3) = false Y axis no hit

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;
}
```

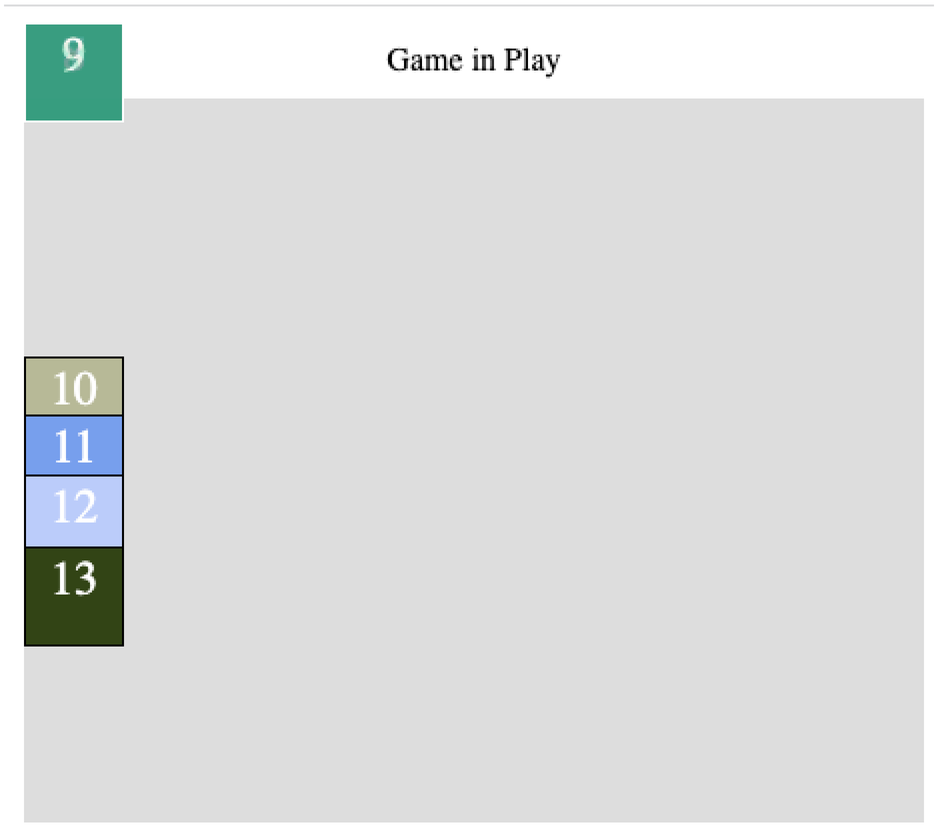
```
        .output2{
            color:black;
            font-size:2em;
        }
        .info div {
            padding: 10px;
        }

        .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);
}
```

Element Catcher Game Part 1 set up

Objective of the game is to click the elements, as they drop, try to land them on the moving platform to score. First lesson is to set up the basic game board and simple interaction with elements and movement.



Use the requestAnimationFrame to create smooth movement between elements on the page.

Create basic interactive elements that move exercise:

1. Create a function that can set up new page elements. Using parameters of the parent element the new element should be added too. The element tag type string name. The inner html content of the element and a class that gets added to the new elements that are created.

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

2. Add an event listener that waits for the DOMContentLoaded to invoke an init function that creates the game board content.
3. Add click events to the new elements
4. Create random background colors for the new element boxes that are created. Math.random().toString(16).slice(2,8)
5. When the element box is clicked add the active class to the element.
6. Create a new function that will handle the animation of elements. This function should be invoked once the start game is pressed, and within itself run the requestAnimationFrame() to the function.
7. Within the mover() function get all the active elements, get the offset top value of the element, and increment it by the game speed value. Apply the new top style position value to the element to make it move.
8. Add a condition to stop the element from moving by removing the active class from the element once it reaches the bottom of the game container.

```
<!DOCTYPE html>

<html>


<head>

   <title>JavaScript Course</title>

   <style>

       * {

           box-sizing: border-box;

       }


       .container {

           position: relative;

           width: 90vw;

           margin: auto;

           background-color: #ddd;

           min-height: 400px;

           text-align: center;
```

```
    overflow: hidden;
}

.message {
    padding: 10px;
    background-color: white;
}

.btn {
    min-width: 50%;
    height: 40px;
    font-size: 1.4em;
    border-radius: 10px;

}

.box {
    position: absolute;
    left: 0;
    top: 0;
    height: 50px;
    width: 50px;
    line-height: 30px;
    text-align: center;
    border: 1px solid white;
    font-size: 1.5em;
    color: white;
    opacity: 0.5;
```

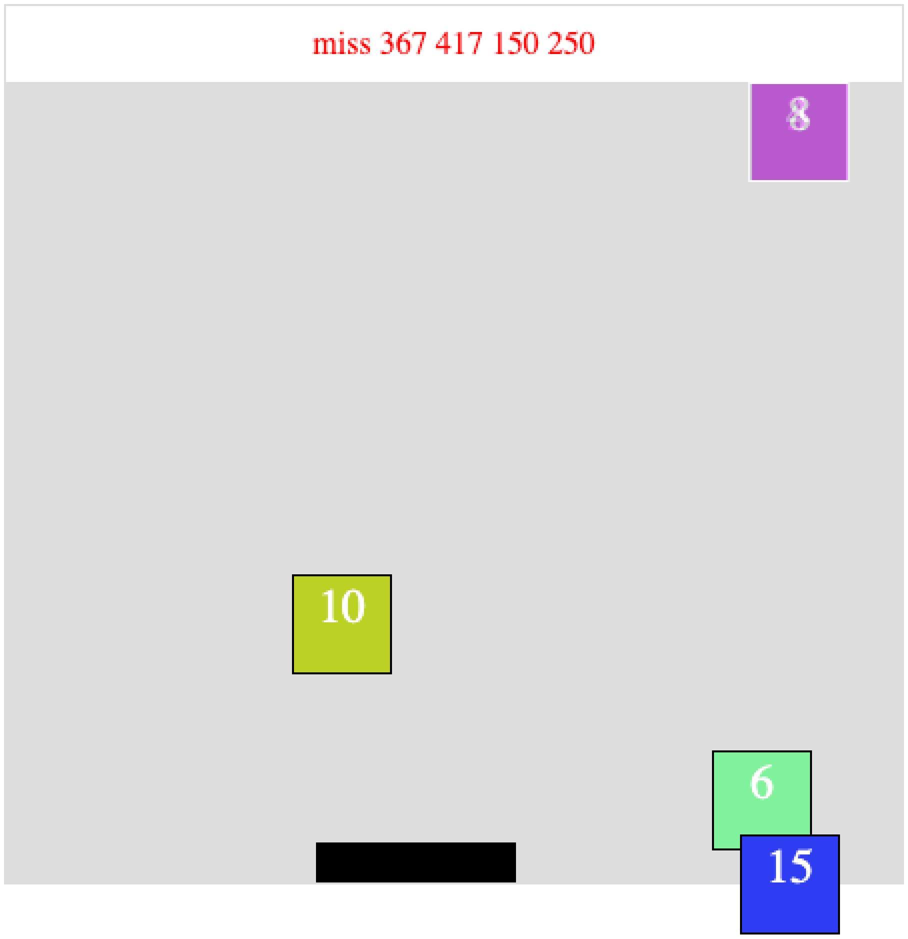
```
        }

        .box:hover {
            cursor: pointer;
            border-color: red;
        }

        .active {
            border-color: black;
            opacity: 1;
        }
    </style>
</head>

<body>
    <div class="container">
    </div>
    <script src="app7.js"></script>
</body>

</html>
```

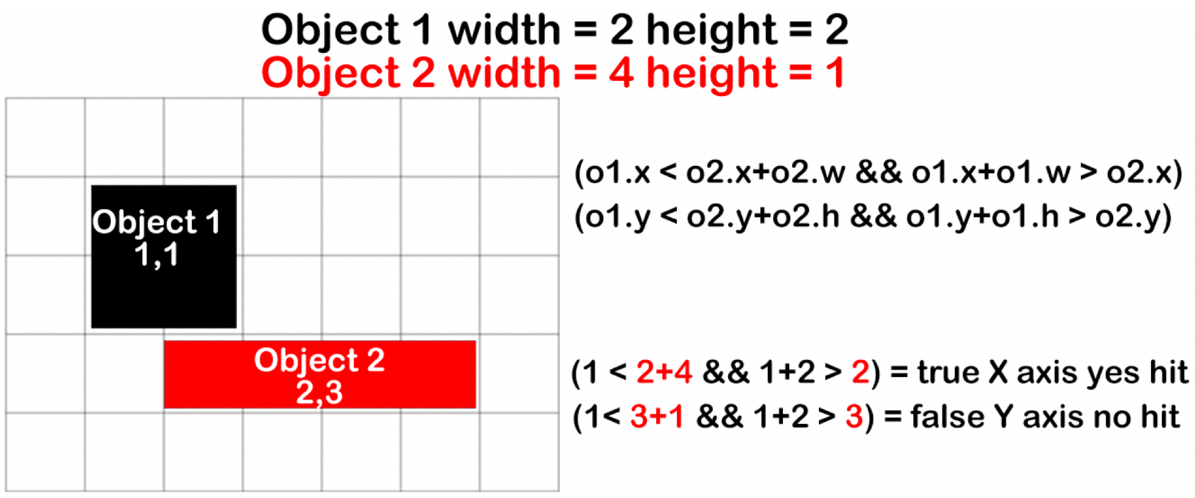
```
const main = document.querySelector('.container');
const message = maker(main,'div','Click Start
Button','message');
const btn1 = maker(main,'button','Start Game','btn');
console.log(message);
const game = {box:[],num:15,ani:{},speed:1};
```

```
btn1.onclick = startGame;
window.addEventListener('DOMContentLoaded',init);

function init(){
   for(let i=0;i<game.num;i++){
       const ele = maker(main,'div',i+1,'box');
       ele.style.backgroundColor =
'#'+Math.random().toString(16).slice(2,8);
       ele.onclick = dropper;
   }
}

function mover(){
   const actives = document.querySelectorAll('.active');
   actives.forEach((ele)=>{
       console.log(ele.offsetTop);
       let y = ele.offsetTop + game.speed;
       if(y > 400){
           ele.classList.remove('active');
       }
       ele.style.top = y + 'px';
   })
   game.ani = requestAnimationFrame(mover);
}


function dropper(e){
   console.log(this);
```

```
    this.classList.add('active');
}


function startGame(){
    btn1.style.display = 'none';
    message.textContent = 'Game in Play';
    mover();
}


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

Element Catcher Part 2 movement and collision detection


miss 367 417 150 250


Object 1 width = 2 height = 2
Object 2 width = 4 height = 1
Object 1
1,1
Object 2
2,3
(o1.x < o2.x+o2.w && o1.x+o1.w > o2.x)
(o1.y < o2.y+o2.h && o1.y+o1.h > o2.y)
(1 < 2+4 && 1+2 > 2) = true X axis yes hit
(1< 3+1 && 1+2 > 3) = false Y axis no hit

Update the game object and detect collision of page elements exercise:

1. Create a paddle element that can move from side to side.
2. Add values in the game object, to be able to stop the game play, and keep track of a score. Create different speeds for the paddle and the

falling box elements. Track the main page width so that elements can be randomly placed within the game screen width.
3. Randomly set the x axis values for the box elements so they are spread out randomly across the top of the screen.
4. Once the paddle goes off the container then set it to move the opposite direction.
5. Add the paddle movement within the animation frame.
6. Within the animation frame check the position of the active elements, checking for both x and y axis overlaps of corners of the elements.
7. If the position of the box plus its height is greater than the paddle y position, and the box y position is less than the paddle plus its height then there is an overlap on the y axis.
8. If there is overlap of the values of the 2 objects, and box x position is less than paddle x plus the width, and also the box plus its width is greater than the paddle x position then there is a hit on the x axis.
9. If a hit occurs then remove the element.
10. If the element ends up having a y position off screen, then reset the random x position and update the y to the top of the screen again.
11. The gameplay will continue removing only the elements that hit the paddle and resetting the misses back to the top.
12. Add tracking of the hits and misses, add the element x,y and the paddle x,y to the message area for debugging if needed.
13. Make adjustments to the position of the page elements to enhance the game screen area.

```
<!DOCTYPE html>

<html>


<head>

   <title>JavaScript Course</title>

   <style>

       * {

           box-sizing: border-box;

       }
```

```
.container {
    position: relative;
    width: 90vw;
    margin: auto;
    border:1px solid #ddd;
    text-align: center;
}
.gameboard {
    position: relative;
    background-color: #ddd;
    min-height: 400px;
}
.message {
    padding: 10px;
    background-color: white;
}

.btn {
    min-width: 50%;
    height: 40px;
    font-size: 1.4em;
    border-radius: 10px;

}
.paddle {
    position: absolute;
    left: 0;
    top: 380px;
    height: 20px;
    width: 100px;
    background-color:black;
}
.box {
    position: absolute;
    left: 0;
    top: 0px;
    height: 50px;
    width: 50px;
    line-height: 30px;
    text-align: center;
    border: 1px solid white;
    font-size: 1.5em;
    color: white;
    opacity: 0.5;
}

.box:hover {
    cursor: pointer;
    border-color: red;
}

.active {
    border-color: black;
    opacity: 1;
}
```

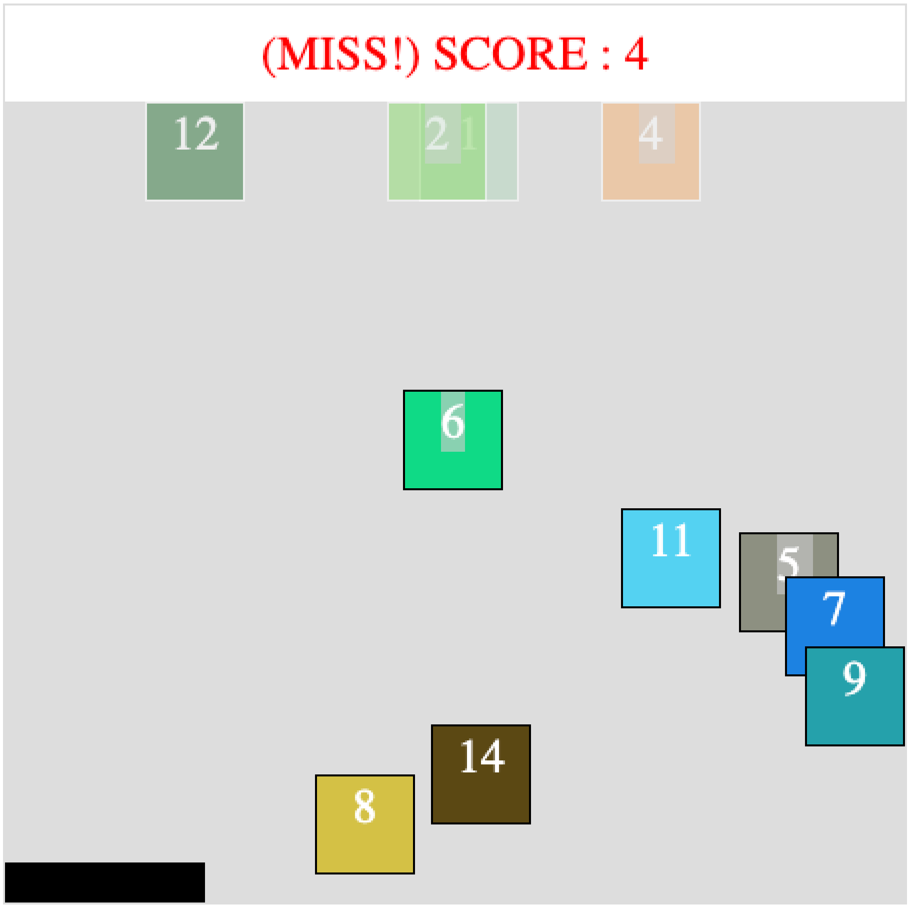
```
   </style>
</head>

<body>
   <div class="container">
   </div>
   <script src="app7.js"></script>
</body>

</html>
```

```
const main = document.querySelector('.container');
const info = maker(main, 'div', '', 'info');
const gameboard = maker(main, 'div', '', 'gameboard');
const message = maker(info, 'div', 'Click Start Button',
'message');
const btn1 = maker(info, 'button', 'Start Game', 'btn');
const paddle = maker(gameboard, 'div', '', 'paddle');
paddle.val = 5;
gameboard.style.display = 'none';
const game = {
   box: [],
   num: 15,
   ani: {},
   pspeed: 1,
   speed: 2,
   width: main.offsetWidth,
   play: false,
   score: 0
};
btn1.onclick = startGame;
window.addEventListener('DOMContentLoaded', init);

function init() {
   for (let i = 0; i < game.num; i++) {
       const ele = maker(gameboard, 'div', i + 1, 'box');
       ele.style.backgroundColor = '#' +
Math.random().toString(16).slice(2, 8);
       ele.onclick = dropper;
       ele.style.left = Math.floor(Math.random() * (game.width -
50)) + 'px';
   }
}

function mover() {
   const actives = document.querySelectorAll('.active');
   actives.forEach((ele) => {
       let y = ele.offsetTop + game.speed;
       ele.style.top = y + 'px';
       if (y > 400) {
           ele.classList.remove('active');
           ele.style.left = Math.floor(Math.random() *
(game.width - 50)) + 'px';
           ele.style.top = '0px';
       }
       const poff = paddle.offsetTop ;
```

```
        if (y + 50 > poff && y < poff + 20) {
            const x = ele.offsetLeft;
            const xend = x + 50;
            const px = paddle.offsetLeft;
            const pxend = px + 100;
            let valHit = 'miss';
            if ((xend > px) && (xend < pxend)) {
                valHit = 'hit';
                message.style.color = 'red';
                ele.remove();
            }
            message.textContent = `${valHit} ${x} ${xend} ${px}
${pxend}`;
            //game.play =false;
        }

    })
    let paddleY = paddle.offsetLeft + (game.pspeed * paddle.val);
    if (paddleY > game.width - 100 || paddleY < 0) {
        paddle.val *= -1;
    }
    paddle.style.left = paddleY + 'px';
    if (game.play) {
        game.ani = requestAnimationFrame(mover);
    } else {
        cancelAnimationFrame(game.ani);
    }
```

```
}

function dropper(e) {
   console.log(this);
   this.classList.add('active');
}


function startGame() {
   game.play = true;
   btn1.style.display = 'none';
   gameboard.style.display = 'block';
   message.textContent = 'Game in Play';
   mover();
}

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

Element Catcher Part 3 Final Game



Game play debugging and final adjustment exercise

1. Play through the game to ensure the proper messaging for the player
2. Add an end game for the game over once all the elements have been landed on the paddle.
3. Provide a way for the player to restart the game, reset the scoring and all the game play to start again.
4. Change the elements to have their own drop speeds, apply various updates to make the game more challenging.

```
<!DOCTYPE html>

<html>


<head>
```

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

    .container {
        position: relative;
        width: 90vw;
        margin: auto;
        border: 1px solid #ddd;
        text-align: center;
    }

    .gameboard {
        position: relative;
        background-color: #ddd;
        min-height: 400px;
    }

    .message {
        padding: 10px;
        background-color: white;
        font-size: 1.5em;
    }

    .btn {
        min-width: 50%;
        height: 40px;
        font-size: 1.4em;
        border-radius: 10px;
        display: block;
        margin: auto;
    }

    .paddle {
        position: absolute;
        left: 0;
        top: 380px;
        height: 20px;
        width: 100px;
        background-color: black;
    }

    .box {
        position: absolute;
        left: 0;
        top: 0px;
        height: 50px;
        width: 50px;
        line-height: 30px;
        text-align: center;
        border: 1px solid white;
        font-size: 1.5em;
        color: white;
        opacity: 0.5;
```

```
        }

        .box:hover {
            cursor: pointer;
            border-color: red;
        }

        .active {
            border-color: black;
            opacity: 1;
        }
    </style>
</head>

<body>
    <div class="container">
    </div>
    <script src="app7.js"></script>
</body>

</html>
```

```
const main = document.querySelector('.container');
const info = maker(main, 'div', '', 'info');
const gameboard = maker(main, 'div', '', 'gameboard');
const message = maker(info, 'div', 'Click Start Button',
'message');
const btn1 = maker(info, 'button', 'Start Game', 'btn');
const paddle = maker(gameboard, 'div', '', 'paddle');
paddle.val = 5;
gameboard.style.display = 'none';
const game = {
   box: [],
   num: 15,
   ani: {},
   pspeed: 1,
   speed: 2,
   width: main.offsetWidth,
   play: false,
   score: 0
};
btn1.onclick = startGame;
//window.addEventListener('DOMContentLoaded', init);

function init() {
   for (let i = 0; i < game.num; i++) {
       const ele = maker(gameboard, 'div', i + 1, 'box');
       ele.style.backgroundColor = '#' +
Math.random().toString(16).slice(2, 8);
       ele.onclick = dropper;
       ele.speedVal = Math.floor(Math.random() * 1) + 1;
       ele.style.left = Math.floor(Math.random() * (game.width -
50)) + 'px';
   }
}
```

```
function mover() {
   const actives = document.querySelectorAll('.active');
   actives.forEach((ele) => {
       let y = ele.offsetTop + ele.speedVal;
       ele.style.top = y + 'px';
       if (y > 400) {
           ele.classList.remove('active');
           ele.style.left = Math.floor(Math.random() *
(game.width - 50)) + 'px';
           ele.style.top = '0px';
       }
       const poff = paddle.offsetTop;
       if (y + 50 > poff && y < poff + 20) {
           const x = ele.offsetLeft;
           const xend = x + 50;
           const px = paddle.offsetLeft;
           const pxend = px + 100;
           let valHit = 'MISS';
           if ((xend > px) && (xend < pxend)) {
               valHit = 'HIT';
               message.style.color = 'green';
               ele.remove();
               game.score++;
           } else {
               message.style.color = 'red';
           }
           message.textContent = `(${valHit}!) SCORE :
${game.score} `;
```

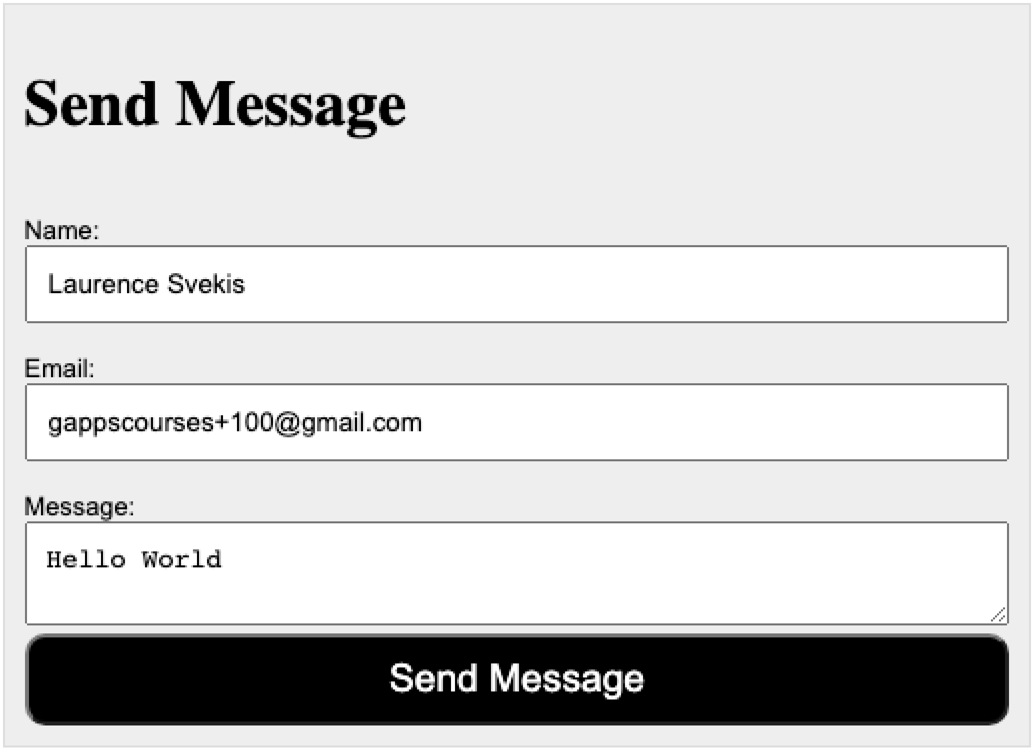
```
            if (game.score >= game.num) {
                //GAME OVER
                game.play = false;
                btn1.style.display = 'block';
                btn1.textContent = 'ReStart';
                message.style.color = 'black';
                message.innerHTML += '<div>Game Over</div>';
            }

            //game.play =false;
        }

    })
    let paddleY = paddle.offsetLeft + (game.pspeed * paddle.val);
    if (paddleY > game.width - 100 || paddleY < 0) {
        paddle.val *= -1;
    }
    paddle.style.left = paddleY + 'px';
    if (game.play) {
        game.ani = requestAnimationFrame(mover);
    } else {
        cancelAnimationFrame(game.ani);
    }

}

function dropper(e) {
    console.log(this);
```

```
   this.classList.add('active');
}


function startGame() {
   game.play = true;
   game.score = 0;
   init();
   btn1.style.display = 'none';
   gameboard.style.display = 'block';
   message.textContent = 'Click the boxes Hit the paddle.';
   mover();
}

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

How to Create a Web Form that can send emails using your Gmail account

HTML Send Email via Apps Script

How to send an email from a HTML form, using Google Apps Script and JavaScript. Front-end code to send emails from html forms.


Send Message

Name:

Laurence Svekis

Email:

gappscourses+100@gmail.com

Message:

Hello World

Exercise : Create an HTML form with fields for the data that you want to send. Setup the JavaScript to validate the form input values, and create an object of data to send to the Apps Script endpoint.

1. Add HTML input fields for the user to be able to enter information. Apply CSS styling as needed to style your form fields
2. Using JavaScript select the input fields as JavaScript variables
3. Create an event that invokes a function named submitter() when the form submit button is clicked
4. Using e.preventDefault(); to prevent the default action of the form submission to prepare for AJAX.
5. Add conditions on the input field values, set up a variable to add content to if errors are caught.
6. If errors are caught in the conditions, output them in a new element that gets created with JavaScript.
7. Set a timeout to remove the error element and reset the input field border colors back to default.
8. Create an object the contains all the form input data with property names that match the data.

HTML CODE

```
<!DOCTYPE html>
<html>

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

    label {
        display: block;
        font-family: Arial, Helvetica, sans-serif;
        font-size: 0.8em;
        padding-top: 15px;
    }

    .myForm {
        width: 90%;
        margin: auto;
        border: 1px solid #ddd;
        padding: 10px;
        background-color: #eee;
    }

    #myForm input,
    textarea {
        width: 100%;
        padding: 10px;

      }

      input[type="submit"] {
        background-color: black;
        color: white;
        text-transform: capitalize;
        font-size: 1.2em;
        border-radius: 10px;
      }
    </style>
  </head>

  <body>
    <div class="myForm">
      <form id="myForm">
        <h1>Send Message</h1>
        <div>
          <label for="name">Name:</label>
          <input type="text" id="name">

        </div>
        <div>
          <label for="email">Email:</label>
          <input type="email" id="email">

        </div>
```

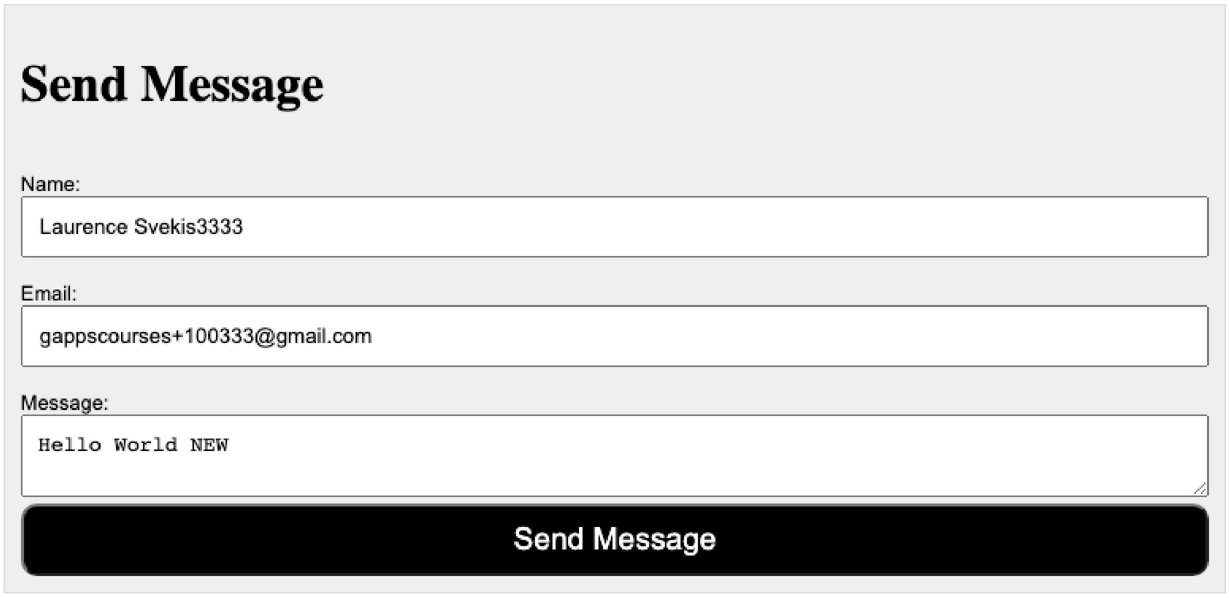
```
            <div>
                <label for="message">Message:</label>
                <textarea id="message"></textarea>

            </div>
            <input type="submit" value="send message">
        </form>
    </div>
    <script src="app3.js"></script>
</body>

</html>
```

JavaScript Code

```
const url = '';
const myForm = document.querySelector('#myForm');
const myName = document.querySelector('#name');
const myEmail = document.querySelector('#email');
const myMessage = document.querySelector('#message');

myName.value = 'Laurence Svekis';
myEmail.value = 'g******courses+100@gmail.com';
myMessage.value = 'Hello World';
myForm.addEventListener('submit', submitter);

function submitter(e) {
    console.log('submitted');
    e.preventDefault();
    let message = '';
    if (myName.value.length < 5) {
        myName.style.borderColor = 'red';
        message += `<br>Name needs to be 5 characters`;
    }
    if (myEmail.value.length < 5) {
        myEmail.style.borderColor = 'red';
        message += `<br>Email is missing`;
    }
    if (message) {
        const div = document.createElement('div');
        div.innerHTML = message;
        div.style.color = 'red';
        myForm.append(div);
        setTimeout(() => {
            div.remove();
            myName.style.borderColor = '';
            myEmail.style.borderColor = '';
        }, 5000);

    } else {
        const myObj = {
            name: myName.value,
            email: myEmail.value,
            message: myMessage.value
        };
        console.log(myObj);
    }
}
```

Send POST request to Google Apps Script

In this lesson the application will be updated to make a fetch request using the POST method to the Google Apps Script endpoint, and insert the form field data into a spreadsheet.



Send Message

Name:

Laurence Svekis3333

Email:

gappscourses+100333@gmail.com

Message:

Hello World NEW

name	**email**	**message**	**status**
Laurence Svekis	g******courses+100@gmail.com	Hello World	success
Laurence Svekis 2	g******courses+5550@gmail.com	Hello World Working ????/	success
Laurence Svekis 2	g******courses+5550@gmail.com	Hello World Working ????/	success
Laurence Svekis3333	g******courses+100333@gmail.com	Hello World NEW	success

Exercise : Update the fetch method to POST, include the form field data as an object in the POST request body contents. Create the Google Apps Script endpoint using a webapp to receive the GET and POST request data from the AJAX request from JavaScript.

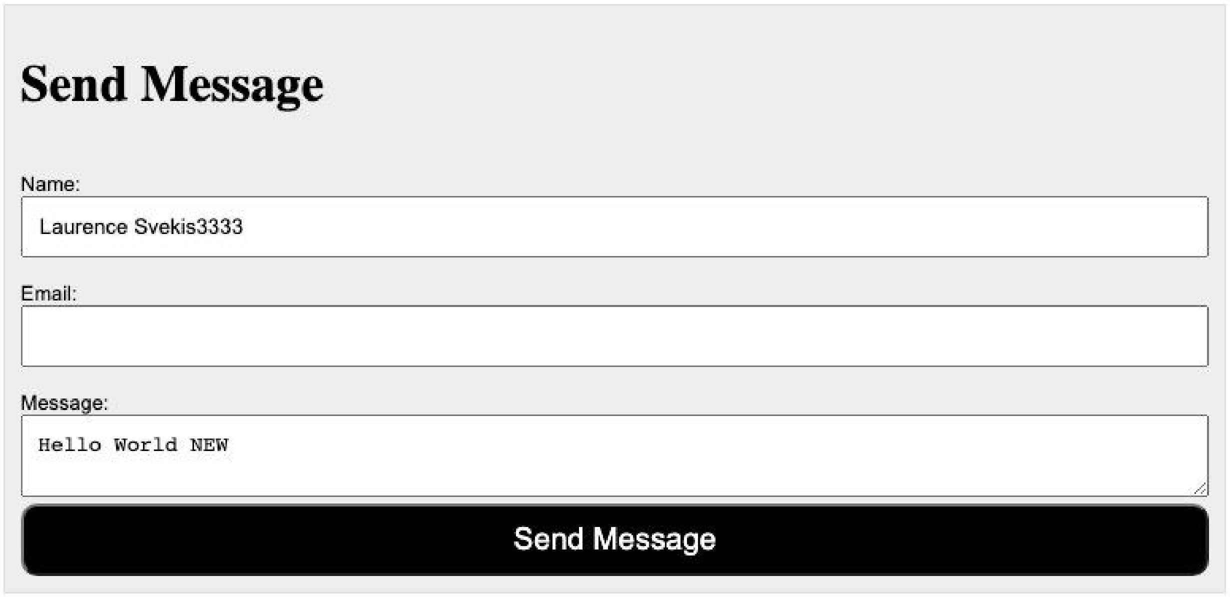
Create a web app with Google Apps Script GET method.

1. Go to https://script.google.com/ create a new project
2. Using the method doGet(e) create a new function that will return the e parameters from the URL request object.

3. Create and return the e parameters as a string value. return ContentService.createTextOutput(JSON.stringify(e))
4. Set the MimeType to JSON setMimeType(ContentService.MimeType.JSON)
5. Deploy a new webapp set the configuration to execute as your account, and who has access to Anyone
6. Copy the web app URL to the JavaScript application as the GET fetch endpoint url.
7. Try sending the fetch request to the Google Apps Script web app and check the response data in the console.

Apps Script Code

```
function doGet(e) {
 return
ContentService.createTextOutput(JSON.stringify(e)).setMimeType(ContentSe
rvice.MimeType.JSON);
}
```



Send Message

Name: Laurence Svekis3333

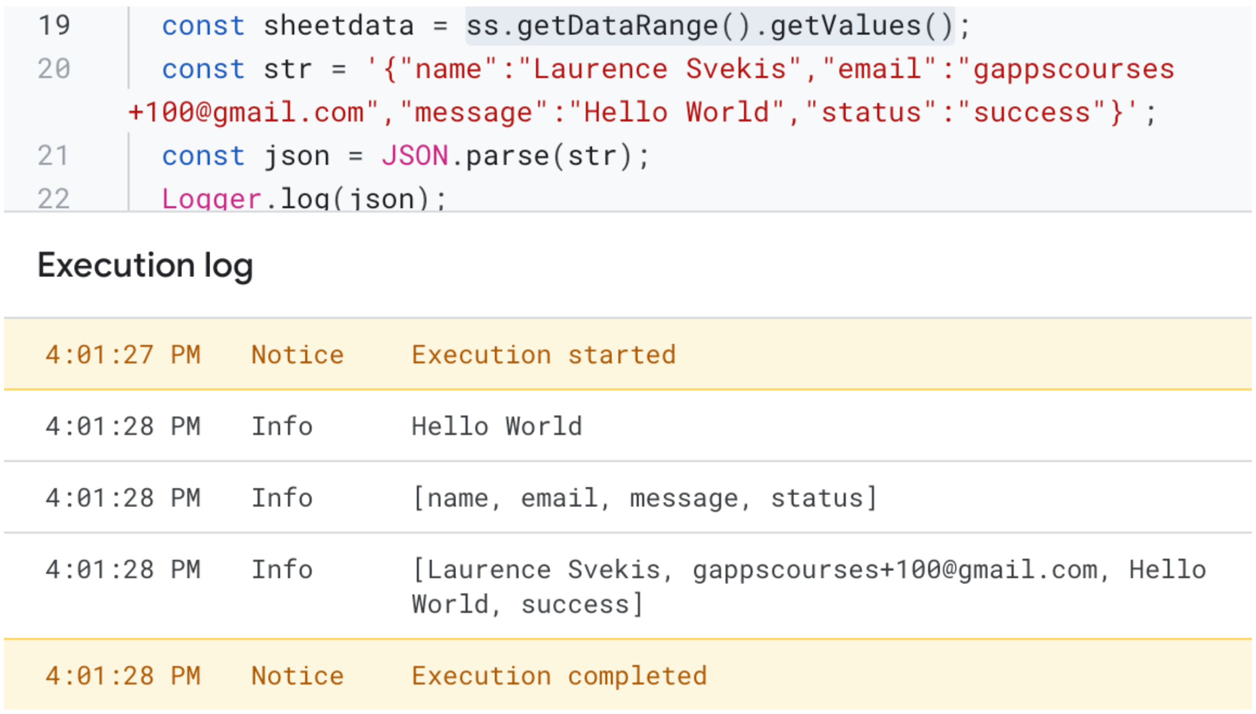
Email:

Message: Hello World NEW

Google Apps Script Testing adding to sheet data

1. Create a new spreadsheet add in the first row the names of the input fields from the HTML form
2. Get the spreadsheet ID from its URL and copy it as the id into the Google Apps Script.
3. Create a function named tester() to be used to test the input of values into the sheet. The doPost() is much harder to troubleshoot so best practice is to get the functionality working first then add it to the doPost() method.
4. Within the testing function open the sheet, SpreadsheetApp.openById(id).getSheetByName('emails') set as a variable
5. Get the data from the spreadsheet, selecting the first row of heading info as an array. ss.getDataRange().getValues()[0]
6. Create an object that can be used for testing with the same property names as the headings in the sheet, and the names from the input fields from the JavaScript object.
7. Loop through the heading names, if there is a value within the data object add that heading using the index value to a temporary holding array.
8. Using appendRow add the holding data to the sheet.
9. Try and run the test application to add to the spreadsheet columns a new row of data.

```
function tester() {
 const id = '1csURUCONXy*****@M-c0';
 const ss = SpreadsheetApp.openById(id).getSheetByName('emails');
 const sheetdata = ss.getDataRange().getValues();
 const str = '{"name":"Laurence
Svekis","email":"gapps*****@.com","message":"Hello
World","status":"success"}';
 const json = JSON.parse(str);
 Logger.log(json);
 Logger.log(sheetdata[0]);
 const holder = [];
 sheetdata[0].forEach((heading, index) => {
   if (json[heading]) holder[index] = (json[heading]);
 })
 Logger.log(holder);
 ss.appendRow(holder);
}
```



```
const sheetdata = ss.getDataRange().getValues();
const str = '{"name":"Laurence Svekis","email":"gappscourses+100@gmail.com","message":"Hello World","status":"success"}';
const json = JSON.parse(str);
Logger.log(json);
```

Execution log

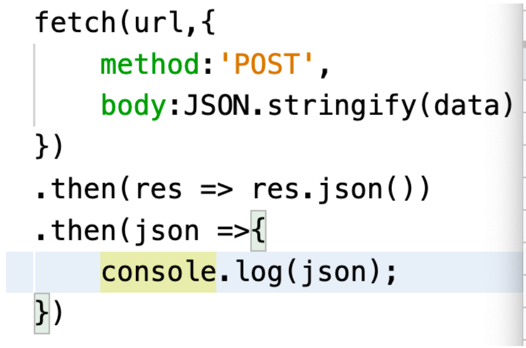
```
4:01:27 PM   Notice   Execution started
4:01:28 PM   Info     Hello World
4:01:28 PM   Info     [name, email, message, status]
4:01:28 PM   Info     [Laurence Svekis, gappscourses+100@gmail.com, Hello
                      World, success]
4:01:28 PM   Notice   Execution completed
```

Create a web app with Google Apps Script POST method.

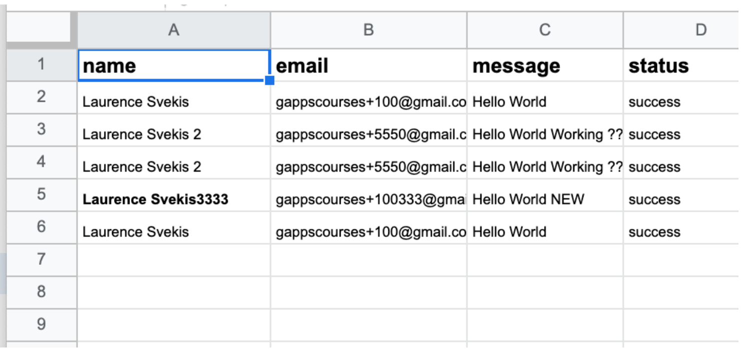
1. Create a new function with the name of doPost(e)
2. Add the testing function code into the doPost()
3. Using the JSON.parse convert the submitted e parameter contents data as a usable JavaScript object. JSON.parse(e.postData.contents)
4. Get the last row of data using the getLastRow() method and add that into the post data contents object.
5. Return the response back to the data.
6. Deploy a web app, you can use the managed deployments to use the existing URL or create a new one and update the URL in your JavaScript code.

7. In the JavaScript code update the AJAX request to the new URL endpoint of the webapp.
8. Send the fetch request as POST with the body of the input field data as a stringified value.
9. Try the code and update the input field values to check the results in the Google Sheet rows.

```
function doPost(e) {
 const id = '1csUR***c0';
 const ss = SpreadsheetApp.openById(id).getSheetByName('emails');
 const sheetdata = ss.getDataRange().getValues();
 const json = JSON.parse(e.postData.contents);
 json.status = 'success';
 const holder = [];
 sheetdata[0].forEach((heading, index) => {
   if (json[heading]) holder[index] = (json[heading]);
 })
 ss.appendRow(holder);
 json.rowval = ss.getLastRow();
 return
ContentService.createTextOutput(JSON.stringify(json)).setMimeType(Content
Service.MimeType.JSON);
}
```



```
fetch(url,{
    method:'POST',
    body:JSON.stringify(data)
})
.then(res => res.json())
.then(json =>{
    console.log(json);
})
```



	A	B	C	D
1	**name**	**email**	**message**	**status**
2	Laurence Svekis	gappscourses+100@gmail.co	Hello World	success
3	Laurence Svekis 2	gappscourses+5550@gmail.c	Hello World Working ??	success
4	Laurence Svekis 2	gappscourses+5550@gmail.c	Hello World Working ??	success
5	**Laurence Svekis3333**	gappscourses+100333@gma	Hello World NEW	success
6	Laurence Svekis	gappscourses+100@gmail.co	Hello World	success
7				
8				
9				

JavaScript Code

```
const url = 'https://script.google.com/macros/s/AKf***C/exec';
const myForm = document.querySelector('#myForm');
const myName = document.querySelector('#name');
const myEmail = document.querySelector('#email');
const myMessage = document.querySelector('#message');

myName.value = 'Laurence Svekis';
myEmail.value = 'gapps*****@@gmail.com';
myMessage.value = 'Hello World';
myForm.addEventListener('submit', submitter);

function submitter(e) {
  console.log('submitted');
  e.preventDefault();
  let message = '';
  if (myName.value.length < 5) {
     myName.style.borderColor = 'red';
     message += `<br>Name needs to be 5 characters`;
  }
  if (myEmail.value.length < 5) {
     myEmail.style.borderColor = 'red';
     message += `<br>Email is missing`;
  }
  if (message) {
     const div = document.createElement('div');
     div.innerHTML = message;
     div.style.color = 'red';
```

```
        myForm.append(div);
        setTimeout(() => {
            div.remove();
            myName.style.borderColor = '';
            myEmail.style.borderColor = '';
        }, 5000);

    } else {
        const myObj = {
            name: myName.value,
            email: myEmail.value,
            message: myMessage.value
        };
        addSendMail(myObj);
    }
}

function addSendMail(data){
    console.log(data);
    fetch(url,{
        method:'POST',
        body:JSON.stringify(data)
    })
    .then(res => res.json())
    .then(json =>{
        console.log(json);
    })
}
```

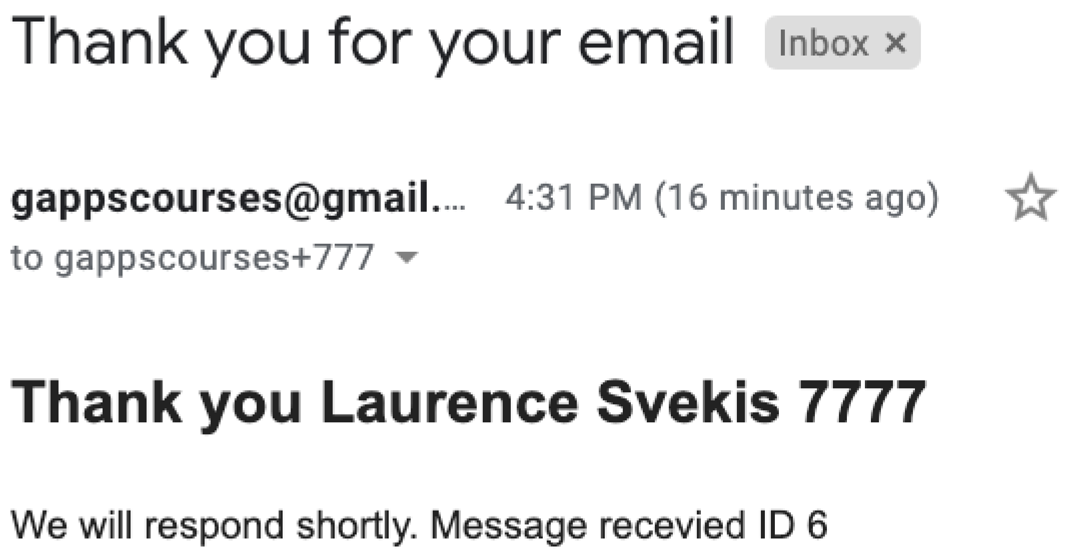
```
function addSendMailGET(data){
   const url1 = url + '?id=100';
   fetch(url1)
   .then(res => res.json())
   .then(json =>{
       console.log(json);
   })
}
```

Google Apps Script Source Code Complete

```
function doPost(e) {
 const id = '1csURUCONX****cR7gM-c0';
 const ss = SpreadsheetApp.openById(id).getSheetByName('emails');
 const sheetdata = ss.getDataRange().getValues();
 const json = JSON.parse(e.postData.contents);
 json.status = 'success';
 const holder = [];
 sheetdata[0].forEach((heading, index) => {
   if (json[heading]) holder[index] = (json[heading]);
 })
 ss.appendRow(holder);
 json.rowval = ss.getLastRow();
 return
ContentService.createTextOutput(JSON.stringify(json)).setMimeType(Content
Service.MimeType.JSON);
}

function tester() {
 const id = '1csURUC*****@gM-c0';
 const ss = SpreadsheetApp.openById(id).getSheetByName('emails');
 const sheetdata = ss.getDataRange().getValues();
 const str = '{"name":"Laurence
Svekis","email":"gap*****@@gmail.com","message":"Hello
World","status":"success"}';
 const json = JSON.parse(str);
 Logger.log(json);
 Logger.log(sheetdata[0]);
 const holder = [];
 sheetdata[0].forEach((heading, index) => {
   if (json[heading]) holder[index] = (json[heading]);
 })
 Logger.log(holder);
 ss.appendRow(holder);
}

function doGet(e) {
 return
ContentService.createTextOutput(JSON.stringify(e)).setMimeType(ContentSe
rvice.MimeType.JSON);
}
```

Send Email when the form is submitted

Send an email to your email address with the form content when the web form is submitted. Send a response confirmation email to the user's email address from the submitted form content.


Thank you for your email Inbox ×

gappscourses@gmail.... 4:31 PM (16 minutes ago)
to gappscourses+777

Thank you Laurence Svekis 7777

We will respond shortly. Message recevied ID 6

Exercise : Update the Google Apps Script to send emails to the user's email address in response to the web form submission, send a second email to your email when the form data is submitted with the form field information.

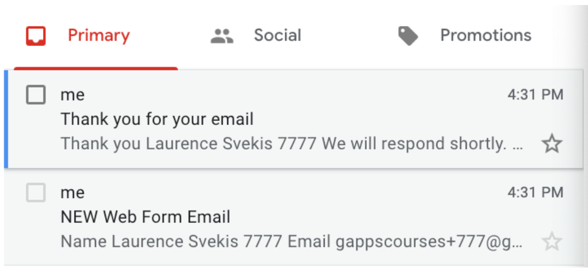
Create a test function to send emails using data from an object

1. Create a function named sendMyEmail that will handle the sending of the emails using an object as the source for the data.
2. Create a function to validate an email address before trying to send an email to that user. This should be included to avoid errors in the Google Apps Script which would result in a CORS issue on the web form.
3. Create a testing function with simulated object data that would be coming from the form. Include the rowval that is set from the sheet row that was inserted.
4. Using the MailApp service, use the sendEmail method to send an email, with an object of parameters for the method. Set the to, subject and htmlBody to the desired values for the email. You should

use the form data structure for the object, to simulate the form submission.
5. Check if the email is valid that the user provided, if it is then using the sendMail send a custom response to the user.
6. Using the test data ensures that you are able to send the email, and accept permissions for the app to use the mailApp service. This is required otherwise the app will not be able to send emails.
7. Move the sendEmail function to the doPost() method, using the real submitted data.
8. Deploy the webapp for the endpoint, if you create a new webapp then ensure you copy the new URL to the web application.

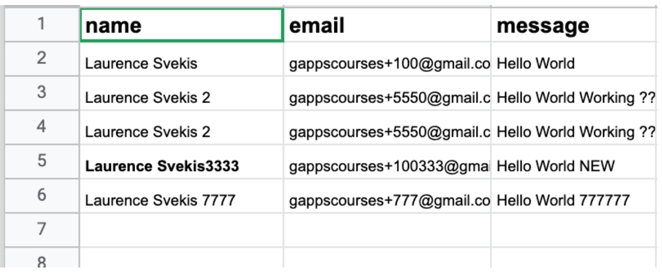
You should be able to send emails to the user, to yourself and also the data should still be added into the spreadsheet whenever the form is submitted.

```
function sendMyEmail(data) {
 let emailBody = `<div>Name ${data.name}</div>`;
 emailBody += `<div>Email ${data.email}</div>`;
 emailBody += `<div>Message ${data.message}</div>`;
 MailApp.sendEmail({
   to: 'g*****@@gmail.com',
   subject: 'NEW Web Form Email',
   htmlBody: emailBody
 });
 if (validateEmail(data.email)) {
   let repHTML = `<h2>Thank you ${data.name}</h2>`;
   repHTML += `<div>We will respond shortly. Message received ID
${data.rowval}</div>`;
   MailApp.sendEmail({
     to: data.email,
     subject: 'Thank you for your email',
     htmlBody: repHTML
   });
   return true;
 } else {
   return false;
 }
}

function validateEmail(email) {
 const re = /\S+@\S+\.\S+/;
 return re.test(email);
}

function testEmail() {
 const val = {
   email: 'gapps*****@gmail.com',
   name: 'tester',
   message: 'Hello World',
   rowval: 50
 }
 Logger.log(sendMyEmail(val));
}
```



me 4:31 PM
Thank you for your email
Thank you Laurence Svekis 7777 We will respond shortly. ...

me 4:31 PM
NEW Web Form Email
Name Laurence Svekis 7777 Email gappscourses+777@g...



1	name	email	message
2	Laurence Svekis	gappscourses+100@gmail.co	Hello World
3	Laurence Svekis 2	gappscourses+5550@gmail.c	Hello World Working ??
4	Laurence Svekis 2	gappscourses+5550@gmail.c	Hello World Working ??
5	**Laurence Svekis3333**	gappscourses+100333@gma	Hello World NEW
6	Laurence Svekis 7777	gappscourses+777@gmail.co	Hello World 777777
7			
8			

Update JavaScript to manage the submission of the data and provide user feedback

1. When the form is submitted, disable the submit button to avoid a second submission. If there are an errors in the form fields enable the button
2. If the form fields are successfully filled out, hide the form element.
3. In the addSendMail() function create a new element, add it to the main content area. Add text for the user to see that their form is submitted.
4. Once a successful response is returned then update the text in the new field with the ID or row value of the imputed content in the sheet. If there was not a success response then show the form for a second submission of data.
5. Make adjustments to the management of the submission process to keep the user informed of the AJAX that the information is sent and the stage of the results from the submission of the data.

JAVASCRIPT

```
const url = 'https://script.google.com/macros/s/AKfyc*******XfK2iR/exec';
const myForm = document.querySelector('#myForm');
const myName = document.querySelector('#name');
const myEmail = document.querySelector('#email');
const myMessage = document.querySelector('#message');
const subBtn = document.querySelector('input[type="submit"]');
const main = document.querySelector('.myForm');
myName.value = 'Laurence Svekis';
myEmail.value = 'gapp*******@gmail.com';
myMessage.value = 'Hello World';
myForm.addEventListener('submit', submitter);

function submitter(e) {
  console.log('submitted');
  e.preventDefault();
  subBtn.disabled = true;
```

```
let message = '';
if (myName.value.length < 5) {
    myName.style.borderColor = 'red';
    message += `<br>Name needs to be 5 characters`;
}
if (myEmail.value.length < 5) {
    myEmail.style.borderColor = 'red';
    message += `<br>Email is missing`;
}
if (message) {
    const div = document.createElement('div');
    div.innerHTML = message;
    div.style.color = 'red';
    myForm.append(div);
    setTimeout(() => {
        div.remove();
        myName.style.borderColor = '';
        myEmail.style.borderColor = '';
    }, 5000);
    subBtn.disabled = false;
} else {
    const myObj = {
        name: myName.value,
        email: myEmail.value,
        message: myMessage.value
    };
    myForm.style.display = 'none';
    addSendMail(myObj);
}
```

```
}

function addSendMail(data){
   console.log(data);
   const repDiv = document.createElement('div');
   repDiv.textContent = 'Waiting.....';
   main.append(repDiv);
   fetch(url,{
       method:'POST',
       body:JSON.stringify(data)
   })
   .then(res => res.json())
   .then(json =>{
       console.log(json);
       if(json.rowval){
           repDiv.textContent = `Message Sent Your ID is ${json.rowval}`;
       }else{
           repDiv.remove();
           subBtn.disabled = false;
           myForm.style.display = 'block';
       }

   })
}

function addSendMailGET(data){
   const url1 = url + '?id=100';
   fetch(url1)
   .then(res => res.json())
   .then(json =>{
       console.log(json);
   })
}
```

Google Apps Script

```
function doPost(e) {
 const id = '1csURUCO********';
 const ss = SpreadsheetApp.openById(id).getSheetByName('emails');
 const sheetdata = ss.getDataRange().getValues();
 const json = JSON.parse(e.postData.contents);
 json.status = 'success';
 const holder = [];
 sheetdata[0].forEach((heading, index) => {
   if (json[heading]) holder[index] = (json[heading]);
 })
 ss.appendRow(holder);
 json.rowval = ss.getLastRow();
 json.result = sendMyEmail(json);
 return
ContentService.createTextOutput(JSON.stringify(json)).setMimeType(Content
Service.MimeType.JSON);
}


function sendMyEmail(data) {
 let emailBody = `<div>Name ${data.name}</div>`;
 emailBody += `<div>Email ${data.email}</div>`;
 emailBody += `<div>Message ${data.message}</div>`;
 MailApp.sendEmail({
   to: 'gap*****gmail.com',
   subject: 'NEW Web Form Email',
   htmlBody: emailBody
 });
 if (validateEmail(data.email)) {
   let repHTML = `<h2>Thank you ${data.name}</h2>`;
   repHTML += `<div>We will respond shortly. Message received ID
${data.rowval}</div>`;
   MailApp.sendEmail({
     to: data.email,
     subject: 'Thank you for your email',
     htmlBody: repHTML
   });
   return true;
 } else {
   return false;
 }
}

function validateEmail(email) {
 const re = /\S+@\S+\.\S+/;
 return re.test(email);
}

function testEmail() {
 const val = {
   email: 'gappsc*****@gmail.com',
   name: 'tester',
   message: 'Hello World',
   rowval: 50
 }
 Logger.log(sendMyEmail(val));
}

function tester() {
 const id = '1csURU*******gM-c0';
 const ss = SpreadsheetApp.openById(id).getSheetByName('emails');
 const sheetdata = ss.getDataRange().getValues();
 const str = '{"name":"Laurence
Svekis","email":"gapp*******@gmail.com","message":"Hello
World","status":"success"}';
 const json = JSON.parse(str);
 Logger.log(json);
 Logger.log(sheetdata[0]);
 const holder = [];
 sheetdata[0].forEach((heading, index) => {
   if (json[heading]) holder[index] = (json[heading]);
 })
 Logger.log(holder);
 ss.appendRow(holder);
}

function doGet(e) {
 return
ContentService.createTextOutput(JSON.stringify(e)).setMimeType(ContentSe
rvice.MimeType.JSON);}
```