SugarCrush Game App -> My replica of one of the world's most famous games CandyCrush mainly using JavaScript. Same to CandyCrush, this replica allows the player to cancel candies whenever they have accumulated three consecutive same-branded candies in row or column, with a score the player getting simultaneously on-side.

Functionalities/Demo:

- A board that randomly generates candies all over the board.
- Allows players to switch out colors of different candies on board.
- Check for 3+ consecutive matches, which are the valid matches to cancel off.
- All the candies get moved down if the player gets a match.

Developing Languages, Tools, and Techniques Needed:

Vscode 1.73 https://code.visualstudio.com/updates/v1_73
JavaScript https://www.javascript.com/
Live Server Vscode Extension v5.7.9 https://www.vsixhub.com/vsix/1950/
CSS3 https://en.wikipedia.org/wiki/CSS
HTML5 https://en.wikipedia.org/wiki/HTML5

Prerequisites & Setups:

Create `index.html`, `app.js` and `style.css` files in local directory to start off.

Synchronous Developing Notes:

Create board:

Create a board of grids in `app.js` and call them:

```
    function createBoard() {
         for (let i = 0; i < width * width; i++) {
             const square = document.createElement('div')
             grid.appendChild(square)
             squares.push(square)
         }
     }
     createBoard()
})
```

Make the board randomly generate different candy colors:

```
  let randomColor = Math.floor(Math.random() * candyColors.length)
  square.style.backgroundColor = candyColors[randomColor]
```

So now we have:

board of randomly generated candy colors.PNG

To make the candy colors grid draggable:

```
square.setAttribute('draggable', true)
```

Now candies colors are draggable:

draggable candy colors.MOV

Drag the candies:

Start to drag:

```
  function dragStart() {
         colorBeingDragged = this.style.backgroundColor
```

```
        squareIdBeingDragged = parseInt(this.id)
        console.log(colorBeingDragged)
        console.log(this.id, 'dragstart')
    }
```

The dragged candies dropped:

```
    function dragDrop() {
        console.log(this.id, 'dragdrop')
        colorBeingReplaced = this.style.backgroundColor
        squareIdBeingReplaced = parseInt(this.id)
        this.style.backgroundColor = colorBeingDragged
        squares[squareIdBeingDragged].style.backgroundColor =
colorBeingReplaced
    }
```

Define valid moves:

```
 if (squareIdBeingReplaced && validMove) {
    squareIdBeingReplaced = null
        } else if (squareIdBeingReplaced && !validMove) {
    squares[squareIdBeingReplaced].style.backgroundColor =
colorBeingReplaced
    squares[squareIdBeingDragged].style.backgroundColor =
colorBeingDragged
        } else squares[squareIdBeingDragged].style.backgroundColor =
colorBeingDragged
```

<u>First check for row of consecutive 3:</u>

If row of 3 is matched, player's score + 3:

```
 function checkRowForThree() {
        for (i = 0; i < 61; i++) {
            let rowOfThree = [i, i + 1, i + 2]
            let decidedColor = squares[i].style.backgroundColor
            const isBlank = squares[i].style.backgroundColor === ''

        if (rowOfThree.every(index =>
squares[index].style.backgroundColor === decidedColor && !isBlank)) {
                score += 3
                rowOfThree.forEach(index => {
                    squares[index].style.backgroundColor = ''
                })
        squares[index].style.backgroundColor = ''
                })
    checkRowForThree()
```

Now we have check row of 3 works:

check row of three works.PNG

Set interval to check for row of three:

```
    window.setInterval(function(){
        checkRowForThree()
     }, 100)
```

Now when manually move to get row of three, it will go into background white color:

row of three works.PNG

First check for column of consecutive 3:

```
    //check column for consecutive 3
     function checkColumnForThree() {
        for (i = 0; i < 47; i++) {
            let columnOfThree = [i, i + width, i + width * 2]
            let decidedColor = squares[i].style.backgroundColor
            const isBlank = squares[i].style.backgroundColor === ''

            if (columnOfThree.every(index =>
squares[index].style.backgroundColor === decidedColor && !isBlank)) {
                score += 3
                columnOfThree.forEach(index => {
                squares[index].style.backgroundColor = ''
```

Now column of 3 matches are checked:

check column of three works.PNG

And manually moving to get a column of 3 candies also works.

Do the same for checking row of 4 and column of 4

Since 4 consecutive has more privileges than 3 consecutive, call 4s before 3s:

```
  window.setInterval(function () {
        checkRowForFour()
        checkColumnForFour()
        checkRowForThree()
        checkColumnForThree()
     }, 100)
```

Get the candies to fall down when there are matches and canceled:

```
    //drop candies once some have been cleared
     function moveDown(){
        for (i =0; i<55; i++ ) {
            if (squares[i+width].style.backgroundColor === ''){
                squares[i+width].style.backgroundColor =
squares[i].style.backgroundColor
                squares[i].style.backgroundColor = ''
            }
        }
...
movedown()
```

Now all the matched candies which are canceled out are all fall down:

matched and canceled candies falls off.PNG

To make the score display:

```
  scoreDisplay.innerHTML = score
```

Add scoreboard to `index.html`:

```
 <div class='score-board'>
        <h3>Score</h3>
        <h1 id="score"></h1>
```

Now we have the scoreboard on-top showing simultaneously:

scoreboard showing.PNG

Import candies images and replace `backgroundColor` with `backgroundImage`:

candies images all showed.PNG