I grew up with a game called “Concentration”; my mom said I would play for hours along with my puzzles. It was a fun game using only a typical deck of cards.

During the days when Flash was part of our curriculum, students needed to learn ActionScript 3.0. The best and most successful activities were game related. I miss those Flash days!

The game you are about to create uses only JavaScript, some ES6 and basic DOM concepts.

Goals

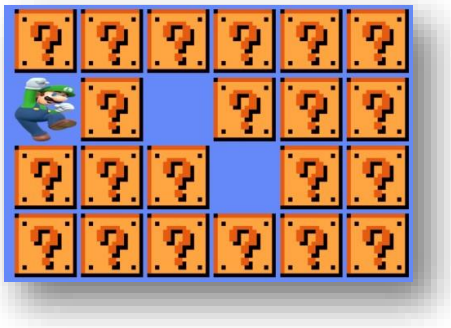
The game has a grid of 24 face-down cards; the card faces consist of pairs of matches. Clicking on cards will flip them over, revealing the value. When two are chosen, if it results in a match, both cards disappear. If the result is not a match, the cards will flip back over to face-down. The game should be different every time the game is refreshed.

Planning

As with any major code problem, it is best to itemize the actual steps. Some developers use flowcharts, others pseudocode; regardless of the method, it helps to break down the code into manageable tasks.

Game Objectives

- Display 12 cards
- Duplicate the cards to have 2 sets of 12
- Randomize the display of cards
- Add selected style for selected cards
- Only allow two cards to be selected at a time
- Determine if two selected cards are a match and hide them
- Reset guess count after 2
- Add delay to selections
- Show back of card initially and flip on select
- Finished game!



Starting Point

Launch the start files for this activity. The file is organized with the following:

- css > style.css (uses a flex grid for the cards)
- images > (images used for the cards)
- js
- index.html

Take time to explore the css file so that you understand the code.

- ***Create a new JavaScript file and then save it as script.js.***

Use the image as a guide to create the code that will display the cards.

First, establish an array for all of the images.

```
'use strict';

let cardsArray = [{
  'name': 'shell',
  'images': 'images/blueshell.png'
}, {
  'name': 'star',
  'images': 'images/star.png'
}, {
  'name': 'bobomb',
  'images': 'images/bobomb.png'
}, {
  'name': 'mario',
  'images': 'images/mario.png'
}, {
  'name': 'luigi',
  'images': 'images/luigi.png'
}, {
  'name': 'peach',
  'images': 'images/peach.png'
}, {
  'name': '1up',
  'images': 'images/1up.png'
}, {
  'name': 'mushroom',
  'images': 'images/mushroom.png'
}, {
  'name': 'thwomp',
  'images': 'images/thwomp.png'
}, {
  'name': 'bulletbill',
  'images': 'images/bulletbill.png'
}, {
  'name': 'coin',
  'images': 'images/coin.png'
}, {
  'name': 'goomba',
  'images': 'images/goomba.png'
}];
```

The cards are all set, now it's time to display them and randomize the display. The result will be something like the following image.

```
38    'images': 'images/goomba.png'
39  }];
40  // Duplicate array to create a match for each card
41  // Randomize the cards
42  let gameGrid = cardsArray.concat(cardsArray);
43  gameGrid.sort(() => 0.5 - Math.random());
44
45  // capture the div with an id of game
46  const game = document.getElementById('game');
47  // create a section with a class of grid
48  const grid = document.createElement('section');
49  grid.setAttribute('class', 'grid');
50  // append the grid section to the game div
51  game.appendChild(grid);
52
53  gameGrid.forEach(item =>
54  {
55  // Create a div
56  const card = document.createElement('div');
57  // Apply a card class to that div
58  card.classList.add('card');
59  // Set the data-name attribute of the div to the cardsArray name
60  card.dataset.name = item.name;
61  // Apply the background image of the div to the cardsArray image
62  card.style.backgroundImage = `url(${item.images})`;
63  // Append the div to the grid section
64  grid.appendChild(card);
65  });
```

Continue to code the script by adding an event listener to the entire grid. Anytime an element is clicked, the selected class will be applied. Also, only two selections at a time are permitted.

Create a new variable.

```
43  gameGrid.sort(() => 0.5 - Math.random());
44
45  let count = 0;
46
47  // capture the div with an id of game
```

```
67  });
68
69  // Add event listener to grid
70  grid.addEventListener('click', function(event)
71  {
72  // The event target is our clicked item
73  let clicked = event.target
74
75  // Do not allow the grid section itself to be selected; only select divs inside the grid
76    if (clicked.nodeName === 'SECTION')
77    {
78      return
79    }
80  // Allow only 2 cards to be selected at a time
81  if (count < 2)
82  {
83      count++;
84      clicked.classList.add('selected');
85    }
86  });
```

In order to test, I added a property to the .selected class.

```
.selected
{
  -webkit-transform: rotateY(180deg);
          transform: rotateY(180deg);
    border: 5px solid purple;
}
```

Continue to code to see if the two selected cards are a match and then hide them. Update the code to add variables.

```
// variables
let firstGuess = '';
let secondGuess = '';
let count = 0;
```

Create a function for matching elements. First, I modified the css for testing purposes.

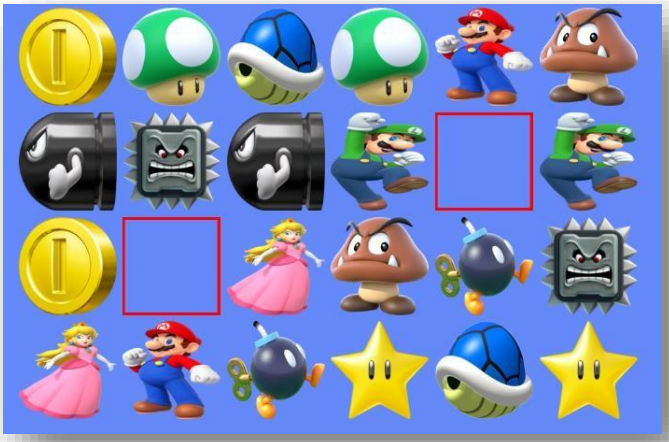
```
.match
{
  border: 5px solid red;
  background: #6589F9 !important;
}
```

Modify the code to make sure of the updates, use the image as a guide.

```
// Add event listener to grid
grid.addEventListener('click', function(event)
{
// The event target is our clicked item
let clicked = event.target

// Do not allow the grid section itself to be selected; only select divs inside the grid
  if (clicked.nodeName === 'SECTION')
  {
    return
  }
// Allow only 2 cards to be selected at a time
if (count < 2)
{
    count++
    if (count === 1)
    {
      // Assign first guess
      firstGuess = clicked.dataset.name
      clicked.classList.add('selected')
    }
    else
    {
      // Assign second guess
      secondGuess = clicked.dataset.name
      clicked.classList.add('selected')
    }
    // If both guesses are not empty...
    if (firstGuess !== '' && secondGuess !== '')
    {
      // and the first guess matches the second match...
      if (firstGuess === secondGuess)
      {
        // run the match function
        match()
      }
    }
  }
})
// Add match CSS
const match = () =>
{
  var selected = document.querySelectorAll('.selected')
  selected.forEach(card =>
  {
    card.classList.add('match')
  })
}
```

If you select 2 cards that match, the css is applied.



Did you select the same element twice? If so, it should have done the following.

Update the code to prevent the selection of the same element, reset the count after 2, and add a delay to the selections.

```
47  // variables
48  let firstGuess = '';
49  let secondGuess = '';
50  let count = 0;
51  let previousTarget = null;
52  let delay = 1200;
53
```

Basically, the image below is the complete js code.

```
'use strict';

let cardsArray = [{
  'name': 'shell',
  'images': 'images/blueshell.png'
}, {
  'name': 'star',
  'images': 'images/star.png'
}, {
  'name': 'bobomb',
  'images': 'images/bobomb.png'
}, {
  'name': 'mario',
  'images': 'images/mario.png'
}, {
  'name': 'luigi',
  'images': 'images/luigi.png'
}, {
  'name': 'peach',
  'images': 'images/peach.png'
}, {
  'name': '1up',
  'images': 'images/1up.png'
}, {
  'name': 'mushroom',
  'images': 'images/mushroom.png'
}, {
  'name': 'thwomp',
  'images': 'images/thwomp.png'
}, {
  'name': 'bulletbill',
  'images': 'images/bulletbill.png'
}, {
  'name': 'coin',
  'images': 'images/coin.png'
}, {
  'name': 'goomba',
  'images': 'images/goomba.png'
}];
// Duplicate array to create a match for each card
// Randomize the cards
let gameGrid = cardsArray.concat(cardsArray).sort(function ()
{
  return 0.5 - Math.random();
});
```

```
46
47  // variables
48  let firstGuess = '';
49  let secondGuess = '';
50  let count = 0;
51  let previousTarget = null;
52  let delay = 1200;
53
54
55  // capture the div with an id of game
56  const game = document.getElementById('game');
57  // create a section with a class of grid
58  const grid = document.createElement('section');
59  grid.setAttribute('class', 'grid');
60  // append the grid section to the game div
61  game.appendChild(grid);
62
63  gameGrid.forEach(function (item)
64  {
65    let name = item.name,
66        img = item.images;
67
68
69    let card = document.createElement('div');
70    card.classList.add('card');
71    card.dataset.name = name;
72
73    let front = document.createElement('div');
74    front.classList.add('front');
75
76    let back = document.createElement('div');
77    back.classList.add('back');
78    back.style.backgroundImage = 'url(' + img + ')';
79
80    grid.appendChild(card);
81    card.appendChild(front);
82    card.appendChild(back);
83  });
84
```

```
84
85  let match = function match()
86  {
87    let selected = document.querySelectorAll('.selected');
88    selected.forEach(function (card)
89    {
90      card.classList.add('match');
91    });
92  };
93
94  let resetGuesses = function resetGuesses()
95  {
96    firstGuess = '';
97    secondGuess = '';
98    count = 0;
99    previousTarget = null;
100
101   let selected = document.querySelectorAll('.selected');
102   selected.forEach(function (card)
103   {
104     card.classList.remove('selected');
105   });
106 };
107
108 grid.addEventListener('click', function (event)
109 {
110
111   let clicked = event.target;
112
113   if (clicked.nodeName === 'SECTION' || clicked === previousTarget || clicked.parentNode.classList.contains('selected') || clicked.parentNode.classList.contains('match'))
114   {
115     return;
116   }
117
118   if (count < 2)
119   {
120     count++;
```

```
121     if (count === 1)
122     {
123       firstGuess = clicked.parentNode.dataset.name;
124       console.log(firstGuess);
125       clicked.parentNode.classList.add('selected');
126     } else
127     {
128       secondGuess = clicked.parentNode.dataset.name;
129       console.log(secondGuess);
130       clicked.parentNode.classList.add('selected');
131     }
132
133     if (firstGuess && secondGuess)
134     {
135       if (firstGuess === secondGuess)
136       {
137         setTimeout(match, delay);
138       }
139       setTimeout(resetGuesses, delay);
140     }
141     previousTarget = clicked;
142   }
143 });
144
```

I hope you enjoyed working through this activity. Why not let your family members play? How about modifying the code so that it provides a message for winning the game, counting the attempts, use different images, etc.