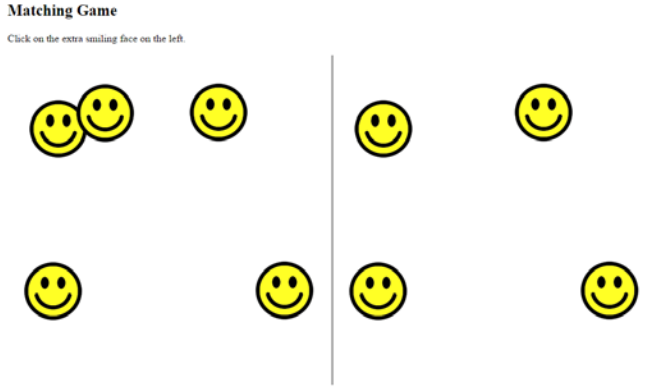
Instructions

Assignment Overview

For this workshop assignment, you will write an image matching game. Please see the accompanying video for a walk-through of gameplay.

Some of the JavaScript skills you will need for this assignment are demonstrated in the Patterns exercise earlier this week in the DOM section. Make sure that you have completed that exercise and understand the code in it, especially the code that is related to DOM manipulation, before you begin this assignment.

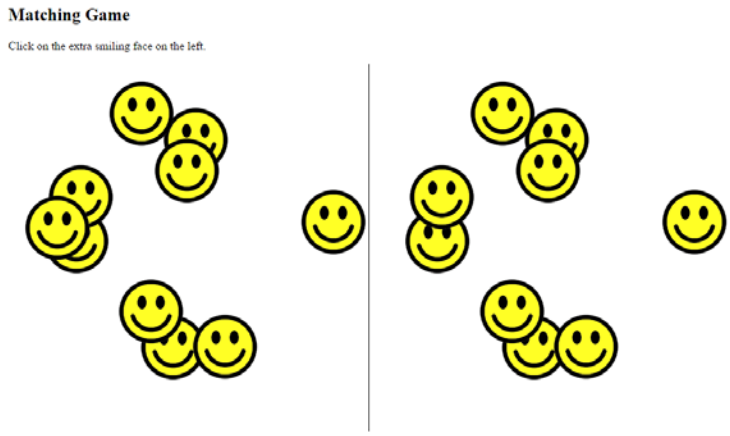
When the game starts, five smiley faces are shown on the left and four are shown on the right. This is illustrated below.



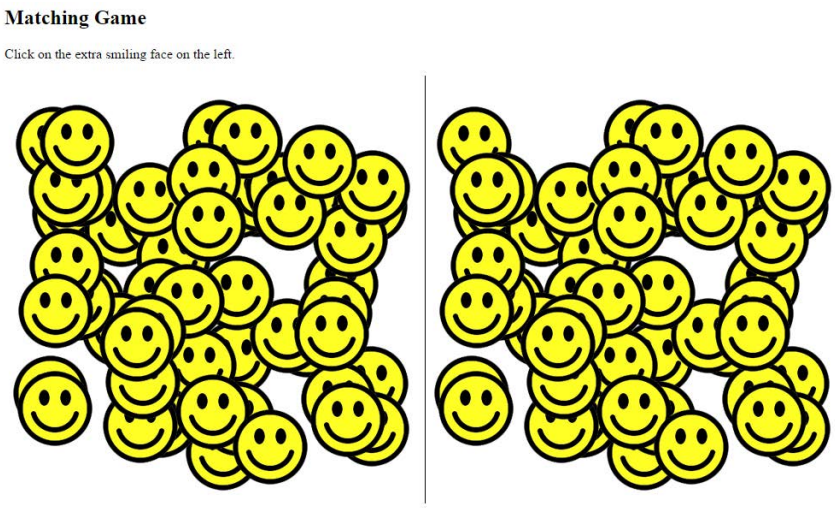
The left and right sides are identical, except for one thing: the left side has one extra face. The user needs to click on that extra face. If anything except the correct face is clicked, a message is displayed saying that the game is over. If the correct face is clicked, all the currently displayed faces are removed and a new set of faces is shown at random positions.

Each time a new set of faces is shown there will be 5 more faces than before, on both the left and the right sides. There will always be one more face shown on the left than on the right. The other faces on the right and left will be identical in position to each other.

For example, let's imagine you are playing the game shown in the previous figure. After clicking on the extra face (top middle) all the faces disappear and the following new set of faces are shown, at new random positions. As you can see, on both sides 5 more faces than before are shown



After playing the game by correctly clicking on the extra face many times, a lot of faces will be shown. This is illustrated below.



Which Browser to Use

This project was developed using the Chrome browser and has been tested to work on Firefox. However, to avoid any potential trouble with inconsistencies between browsers, it may be wise for you to use Chrome.

Assignment Tasks

To help you organize your approach, detailed instructions are provided below in the form of 4 main tasks, each separated into smaller parts.

Summary of Task 1: Set up the HTML and CSS.
Summary of Task 2: Write the JavaScript for the left side of the game.
Summary of Task 3: Write the JavaScript for the right side of the game.
Summary of Task 4: Finish the game logic.
Add the smiley image to your **images** folder. Transparent (gif, tif, png)
https://learn.nucamp.co/pluginfile.php/619/mod_assign/intro/smile.png
http://drpage.net/Nucamp/Images/New_Yr.png
http://drpage.net/Nucamp/Images/penny.png

https://www.lunapic.com

TASK 1: Set up the HTML and CSS

For this task, you will create the HTML page without any JavaScript. The result will look like the below screenshot when viewed in a browser. As you can see, only the instructions and the middle line are visible. *After viewing... remove the image.*

CSS Position

static: Default value. Elements render in order, as they appear in the document flow
fixed: The element is positioned relative to the browser window
absolute: The element is positioned relative to its first positioned (not static) ancestor element
relative: The element is positioned relative to its normal position, so "left:20px" adds 20 pixels to the element's LEFT position
sticky: like static, but scrolls until it reaches an offset that you set

Part 1: Set up the HTML

- Create a basic valid HTML5 document as you learned to do in Week 1, and give it the name **matching-game.html**. Be sure to set the **DOCTYPE**, the **meta charset**, and an appropriate **title** (such as 'Matching Game').
  - In the **body** element, add an **h1** element with the text content of **Matching Game**.
    Beneath this, add a button element with the "type" of "button", add an onclick attribute with the value of "generateFaces()", and text content of "Start Game" (without the quotes).
- *Add text with the instructions e.g.*
  'Click on the extra smiling face on the left.'

Part 2: Add left and right divs (and add the <script> tags)

- You will also need 2 divs (id="leftSide", and id="rightSide")

Also add 1 HTML container 'main' (id="board") element if you use the button

You might just want to add one image as a test...

```
<div id="leftSide"> <img src="../images/smile.png" style="top: 100px; left: 100px;"> </div>
```

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
        content="width=device-width, initial-scale=1.0">
    <title> The Smiley Game</title>
</head>
<body>
    <h1> The Smiley Game </h1>
    <button onclick="generateFaces()">Start Game</button>
    <p>Click on the extra smiling face on the left.</p>
    <main id="board">
        <div id="leftSide"> add test image</div>

        <div id="rightSide"></div>
    </main>
<script>
   //PART 2  AND  3 GO HERE
</script>
</body>
</html>
```

Part 3: Add CSS

- Use an internal stylesheet to add the following styles.
  1. For all **img** elements:
     ```
     position: absolute;
     ```
     This is so that we can fix the exact position of any image later. Although we haven't actually added any images to the webpage at this stage, we are adding this style rule now so that we can easily control their exact position later.
  2. For all **div** elements:
     ```
     position: absolute;
     width: 500px;
     height: 500px;
     ```
     This will set both the left and right divs to be 500px square each, with absolute positioning.
  3. For **only** the div with the id of **rightSide** (recall how to use an ID selector):
     ```
     left: 500px;
     border-left: 1px solid;
     ```
     This moves the rightSide div 500 pixels to the right, so that it is to the right of the leftSide div. Then it uses the **border-left** property to create a vertical line between the two divs.

Feel free to tweak the border and positions…

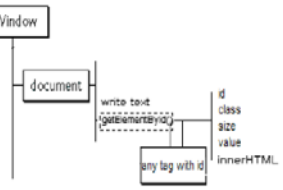
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
        content="width=device-width, initial-scale=1.0">
    <title>Matching Game</title>
    <style>
        img{position: absolute;}
        div{
            position: absolute;
            width: 500px;
            height: 500px;
        }
        #leftSide{border: blue 1px solid;}
        #rightSide{
            left: 511px;
            border: blue 1px solid;
        }
    </style>
</head>
```

TASK 2: Generate the left side images - Alternate Approach. (and one last look at the DOM)

For Part 2: [Generating the left side images]

- For this task you need to add **JavaScript** code which generates (a variable called) *numberOfFaces* images on the left side. (There are no right side faces in part 2).
- The result will look like this when viewed in a browser.

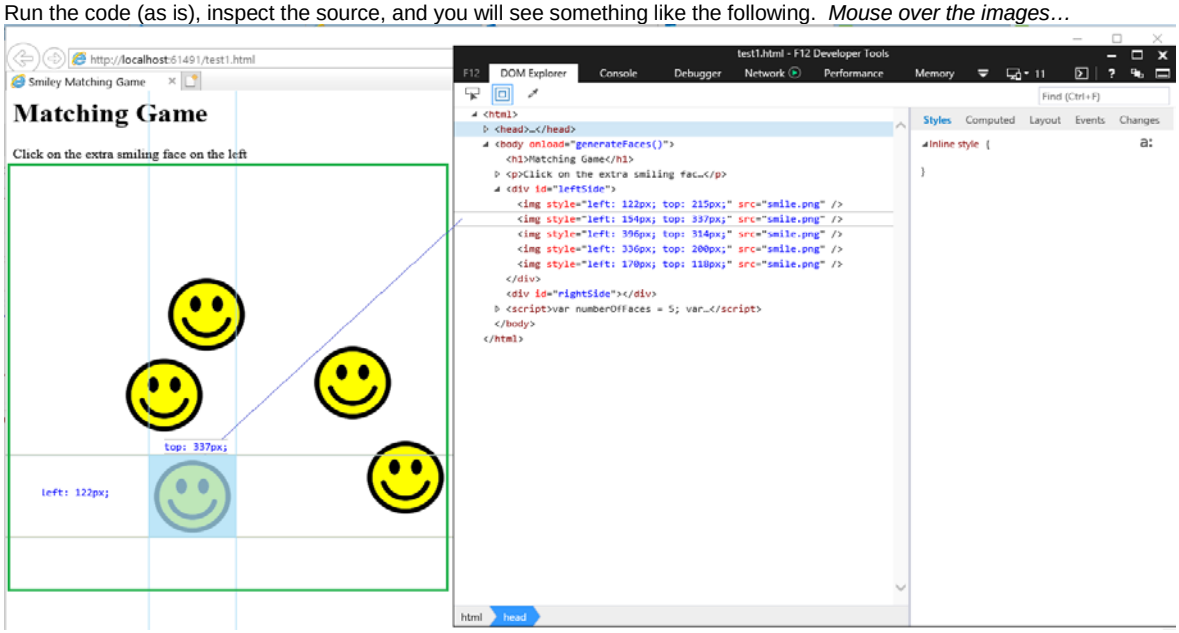

```
NOTE: YOU NEED AN ONLOAD/ONCLICK EVENT TO MAKE THIS WORK:
onload="generateFaces()"> or onclick="generateFaces()"

let numberOfFaces  = 5;
let theLeftSide = document.getElementById("leftSide");
let theRightSide = document.getElementById('rightSide');

//theLeftSide var now controls all properties/methods of the div element with the id "LeftSide"
function generateFaces() {
    for (i = 1; i <= numberOfFaces; i++) {  // standard for loop
        let randomTop = Math.floor(Math.random() * 400);
        let randomLeft = Math.floor(Math.random() * 400);
        let face = document.createElement("img");
            face.src = "smile.png";
            face.style.top = randomTop + "px";
            face.style.left = randomLeft + "px";
         theLeftSide.appendChild(face);
        }
//more code
    }
Think of the variable face as a dynamic way to create the html tag:
        <img  src="smile.png" style="top: 100px; left: 100px;">
```



Remember from last week: Random Numbers

```
Math.random();      // returns a random number from 0 to .99999999999
Math.floor():      // rounds a number down to the lowest whole number
Math.floor(Math.random() * 400); // returns a random integer from 0 to 399


This is the updated code:
            let randomTop = Math.floor(Math.random() * 400);
            let randomLeft = Math.floor(Math.random() * 400);
```

Run the code (as is), inspect the source, and you will see something like the following. *Mouse over the images…*

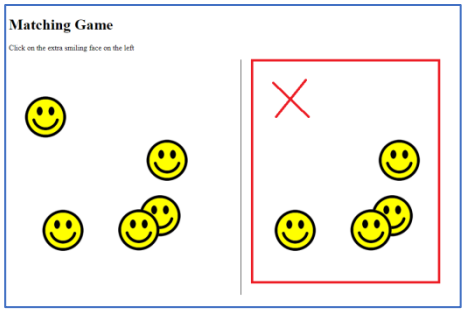


TASK 3: Handle the right side

A couple more DOM methods -
The **createElement()** method is used to create any HTML element. Then you add attributes and style to it…

```
var face = document.createElement('img');
```

The **removeChild()** method removes a specified child node of the specified element..
removeChild(*node*) has 1 parameter: fistChild, lastChild, etc…



You will now extend the JavaScript you developed in Task 2 to generate images on the right side. At the end of this task, your assignment should look similar to this in the browser, with smiley faces on both sides, one less on the right side:

The cloneNode() method creates a copy of a node, and returns that clone.
cloneNode(true/false) uses only 1 parameter: true (false wouldn't do anything)
true => Clones the node, its attributes, and its descendants…

Part 1: Global variable

- Declare a new global variable named **theRightSide**, below where you declared the variable **theLeftSide.** Point this variable to the div with the ID of **rightSide**. **ALREADY DONE**

Part 2: Clone images to the right side

- Add the following code in the **generateFaces** function, below the for loop you created in Task 2:
  - Declare a variable called **leftSideImages** or similar. For its value, clone the entire **theLeftSide** div node, including all its children, i.e.:

```
let leftSideImages = theLeftSide.cloneNode(true);
```

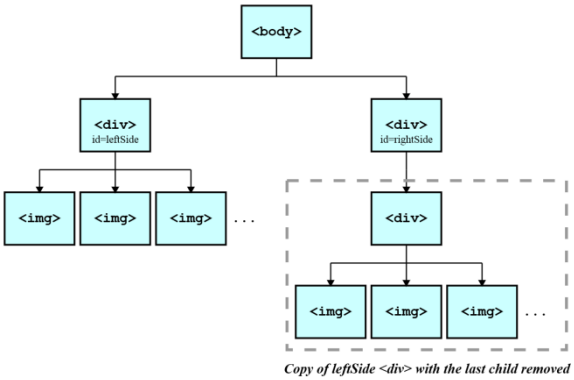
  - After this, use **removeChild()** to remove the last child of **leftSideImages**. This will ensure that there is one less smiley face on the right side.
  - Use **appendChild()** to append **leftSideImages** as a child of **theRightSide**.

```
Already done




---

After the for loop… but before closing function tag… add the following

        }


        let leftSideImages = theLeftSide.cloneNode(true);




        leftSideImages.removeChild(leftSideImages.lastChild);

        theRightSide.appendChild(leftSideImages);
}
```

A couple of DOM methods -
The **appendChild()** method is used to create a node as the last child of the node (used for creating a new element).
theLeftSide.appendChild(thisSmiley);

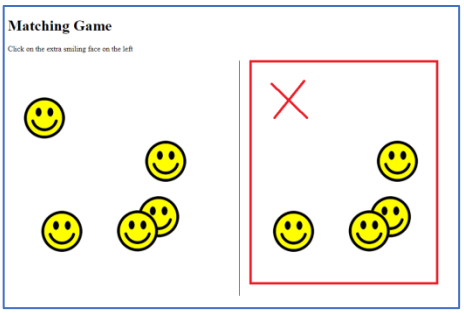
The **cloneNode()** method creates a copy of a node, and returns that clone.
cloneNode(true/~~false~~) uses only 1 parameter: true
true - Clone the node, its attributes, and its descendants

At this point, the DOM will look like this:



Copy of leftSide with the last child removed

check what you see in your browser



What does this means in terms of the DOM and JavaScript?… See MyDOM.html (Create an entire HTML site using DOM methods)

Part 3: Load the Function

- To call the **generateFaces** function when the webpage is loaded
- **REM:** You already have an **onclick** event (task 1.part 1)

```
<body onload="generateFaces()"   or
<button onclick="generateFaces()"
```

At this point, you should be able to view your webpage in your browser and see that it looks like the screenshot shown above for Task 2.

```
<body>

    <h1> The Smiley Game </h1>
    <button onclick="generateFaces()">Start Game</button>
    <p>Click on the extra smiling face on the left.</p>
    <div id="board">
        <div id="leftSide"></div>
        <div id="rightSide"></div>
    </div>
    <script>
        //
        let numberOfFaces = 5;
        const theLeftSide = document.getElementById('leftSide');
        const theRightSide = document.getElementById('rightSide');

        function generateFaces() {
            for (let i = 0; i < numberOfFaces; i++) {
                let face = document.createElement('img');
                    face.src = '../Images/smile.png';
                const randomTop = Math.floor(Math.random() * 400) + 1;
                const randomLeft = Math.floor(Math.random() * 400) + 1;
                face.style.top = randomTop + 'px';
                face.style.left = randomLeft + 'px';
                theLeftSide.appendChild(face);
            }

            const leftSideImages = theLeftSide.cloneNode(true);
            leftSideImages.removeChild(leftSideImages.lastChild);
            theRightSide.appendChild(leftSideImages);

        }

    </script>
</body>
```

- The **cloneNode()** method creates a copy of a node, and returns that clone.
  cloneNode(true/~~false~~) uses only 1 parameter: true (false wouldn't do anything)
  true - Clone the node, its attributes, and its descendants

Test your code

TASK 4: Finish the game logic

In this task, you will complete the JavaScript code developed in Tasks 2 and 3 to include adding and removing event handlers and other game logic.

Part 1: Add an event handler function to the extra face using addEventListener

The player is supposed to click on the extra smiley face on the left side that is ***not*** on the right side. Remember, the faces on the right side are a clone of the left side, except for the last child which was been removed on the **theRightSide** node.

That means the extra smiley face on the left side will be the last child of the **theLeftSide** node. You will need to add an event handler to this face so that when it is clicked, the program will go to the next level.

- Inside the **generateFaces** function, below the existing code, set up an event listener for the **last child** of **theLeftSide** with **addEventListener**.
- Have **addEventListener()** listen for the event '**click**' as the first argument. For the second argument, use the function name **nextLevel** (or something similar).

The **addEventListener(*event, function*)** method attaches an event handler to a specified element. It has 2 required parameters:

event - Required. A String that specifies the name of the event. (i.e., *click*, mouseover, keydown, mouseout)

function - Required. Specifies a function to run when the event occurs. *No parenthesis*

```
    theLeftSide.lastChild.addEventListener('click', nextLevel);
    document.getElementById("board").addEventListener('click', gameOver);
// the gameOver() function is included at the end
} //End of generateFaces function
```

Add both functions

```
function nextLevel(event) {
    alert("You win");
}
function gameOver() {
    alert("You Loose");
}
```

Notice Bubbling…

- Outside of the function **generateFaces**, declare a new function named **nextLevel** (or whatever name you used for the second argument in the event listener you just created.)
- The starting code for this function is provided below:

```
function nextLevel(event) {
    event.stopPropagation();
    numberOfFaces += 5;
    generateFaces();
}
```

- Add the command `event.stopPropagation();`
  This is necessary in order to ensure that the event does not also get applied to other elements in the web page, such as other faces. That would trigger the function multiple times, which is not what we want.

Add a while loop to clear out the original images on both sides… Otherwise you will just add additional images on top of what is already there…
You can use the removeChild method to do this, looping until there are no more children

OR

With two lines of code you can delete all images from both sides

```
theLeftSide.innerHTML = '';
theRightSide.innerHTML = '';
```

- Add a line `numberOfFaces += 5;`
  increases the number stored in numberOfFaces by 5,
  so that the next time the faces are generated there are 5 more than before on both sides
- Add a line `generateFaces();` Basically, this restarts the game,
  And means that a new set of faces is generated. Because of the increase in value of **numberOfFaces**, there will be 5 more faces than before on both sides.

```
function nextLevel(event) {
    event.stopPropagation();

    while (theLeftSide.firstChild) {
        theLeftSide.removeChild(theLeftSide.firstChild);
    }
    while (theRightSide.firstChild) {
       theRightSide.removeChild(theRightSide.firstChild);
    }

    numberOfFaces += 5;

    generateFaces();

} //End of nextLevel function
```

The stopPropagation() method prevents propagation of the same event from being called. It prevents further spread of the current event. See Below… (Prevents any propagation of the same event)

Event Propagation

Event propagation is a blanket term for both **event** *bubbling* and **event** *capturing*.

Consider the page we have built so far…

We have images inside of divs, inside of other divs, not to mention the body and html…

Our DOM Model (propagation path) looks something like this…

window,
document,
HTML,
BODY,
DIV, (main)
DIV, (rightSide)
DIV, (leftSide)
IMG

A click on an image does not only generate a click event for the corresponding IMG element, but also for the parent DIV, for the grandfather DIV and so on...

In our case we really only care about the IMGs in the in the leftSide, DIVs… But the propagation is still across all the nodes in the propagation path.

So, the event propagation can be stopped in any event listener by invoking the **stopPropagation** method of the event object. This means that all the listeners registered on the nodes on the propagation path that follow the current target will not be called.

Notes:
With **bubbling**, the event is first captured and handled by the innermost element and then propagated to **outer** elements.

With **capturing**, the event is first captured by the outermost element and propagated to the **inner** elements.

Part 2: Add an event handler function to the body using addEventListener() ALREADY DONE

- You need to add another function for handling the situation when the player clicks on anything except the correct face.
- In the **generateFaces** function, below where you added the last event listener, use **addEventListener()** one more time. You will use it on **document.getElementById("board")** if using a button with the first argument **'click'** and the second argument of **gameOver**.
- Outside of the **generateFaces** function, declare a third function named **gameOver** with an empty parameter list.
- Inside this function, write an **alert()** with the message of 'Game Over!'.

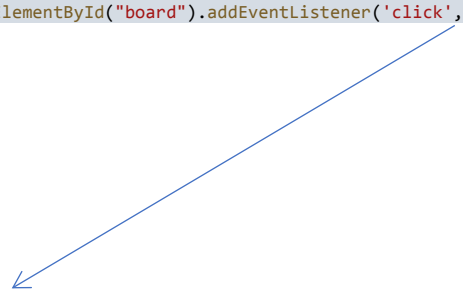
Part 3: Remove event handler functions using removeEventListener()

- Inside the **gameOver** function, use **removeEventListener** twice to remove the event handlers you created with **addEventListener**. That means **document.getElementById("board")** if using a button to remove the **gameOver** event handler function for the **'click'** event, and on **theLeftSide.lastChild** to remove the **nextLevel** event handler function for the **'click'** event.

Part 4: Delete the child nodes

- Each time the player clicks on the correct face, all faces must be removed before a new set of faces is generated and added to the page. So that means at the appropriate place, **all children of both theLeftSide and theRightSide div nodes need to be removed**. In the past week, you learned how to remove all child nodes of a parent node using a **while** loop. Use that knowledge to add **two while loops** to the **nextLevel** function to remove all child nodes of **theLeftSide** and **theRightSide.** Be sure to place these loops **before** the call to **generateFaces()** in the **nextLevel** function.
- Don't forget to reset the number of images back to 5.

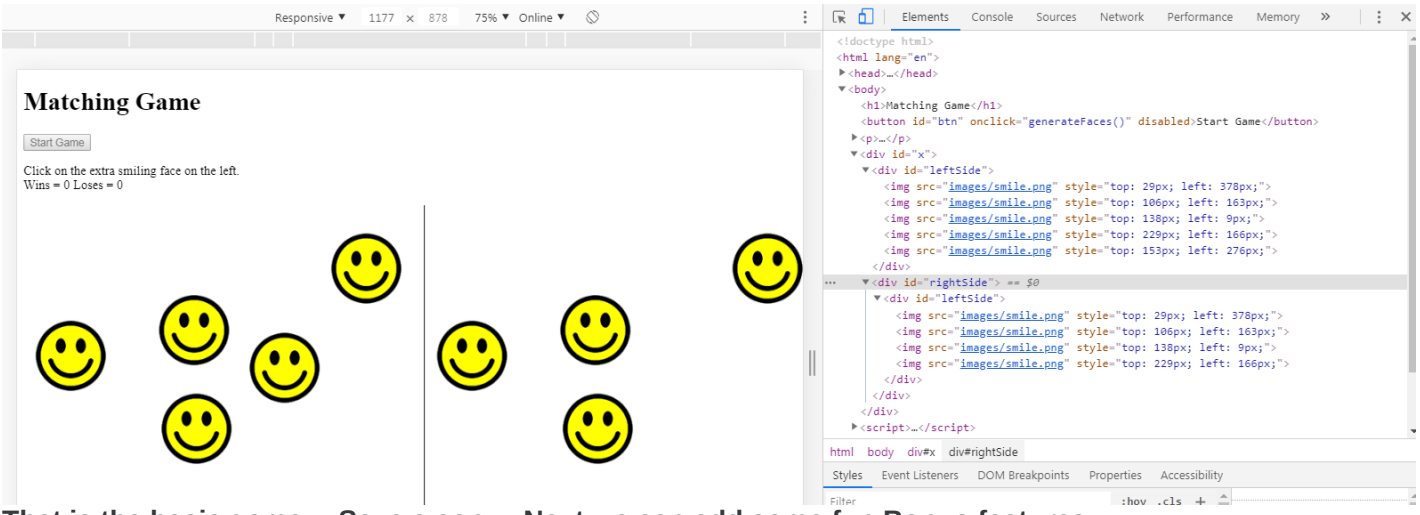
```
  theLeftSide.lastChild.addEventListener('click', nextLevel);
  document.getElementById("board").addEventListener('click', gameOver);
```



```
function gameOver() {
    alert('You Lose!');

document.getElementById("board").removeEventListener('click', gameOver);
theLeftSide.lastChild.removeEventListener('click', nextLevel);

    theLeftSide.innerHTML = '';  //Alternative to removeChild loop
    theRightSide.innerHTML = '';
```

```
while (theLeftSide.firstChild) {
  theLeftSide.removeChild(theLeftSide.firstChild);
}
while (theRightSide.firstChild) {
  theRightSide.removeChild(theRightSide.firstChild);
}
```

```
    numberOfFaces = 5;
}
```



That is the basic game… Save a copy. Next we can add some fun Bonus features…

BONUS CHALLENGES

When the game ends, show a button that will allow the player to restart the game.

Already Done

```
document.querySelector("#btn").setAttribute("disabled", true);
document.querySelector("#btn").removeAttribute("disabled");
```

```
<button id="btn" onclick="generateFaces()">Start Game</button>

   function generateFaces() {
         document.getElementById("btn").disabled = true;

   function gameOver() {
          document.getElementById("btn").disabled = false;
```

But you can disable the button, so that the player can't click it until the game ends. *Where else would you use this?*

Add a counter that shows how many attempts the player has made, and show them the count at the end of the game.

```
 <p>Click on the extra smiling face on the left.<br>
    Wins = <span id="win">0</span> Loses = <span id="los">0</span>
 </p>

        let wins = 0; //Global Variable
        let loss = 0; //Global Variable

        function nextLevel(event) {
            wins = wins + 1;
            document.getElementById("win").innerHTML = wins;

        function gameOver() {
            loss = loss + 1;
            document.getElementById("los").innerHTML = loss;
```

Allow the user to **Choose how many additional faces** are generated for each level, i.e. allow them to set the numberOfFaces variable (through a prompt or similar) at the beginning of the game.

Alternatively, allow the user to choose a mode at the beginning of the game (e.g. Easy, Normal, Difficult) then set the numberOfFaces accordingly (e.g. 2 for easy, 5 for normal, 8 for difficult...)

```
    <p>Click on the extra smiling face on the left.<br>
    Wins = <span id="win">0</span> Loses = <span id="los">0</span>
    <br>How many faces do you want to generate:
      <input id="numFace" type="number" value="1" style="width: 40px">
    </p>
=-=-=-=-=-=-=-=-=-=-=-=-=-=-
<script>
    let wins = 0; //Global Variable
    let loss = 0; //Global Variable
     let numberOfFaces; // = 5; this is dynamic - increases with wins
     let numFace;  // static - never changes -
     let newStart = true; // check for new game
    const theLeftSide = document.getElementById("leftSide");
    const theRightSide = document.getElementById('rightSide');
=-=-=-=-=-=-=-=-=-=-=-=-=-=-

    function generateFaces() {
        if(newStart == true){
          numberOfFaces = document.getElementById('numFace').value;
          numFace = document.getElementById('numFace').value;
        }
=-=-=-=-=-=-=-=-=-=-=-=-=-=-

    function nextLevel(event) {
      wins = wins + 1;
      document.getElementById("win").innerHTML = wins;
        newStart = false;
      event.stopPropagation();
      while (theLeftSide.firstChild) {
          theLeftSide.removeChild(theLeftSide.firstChild);
      }
      theRightSide.innerHTML = ''; //alturnative to remove child loop
        numberOfFaces = Number(numberOfFaces) + Number(numFace); //5;
      generateFaces();
    } //End of nextLevel function

=-=-=-=-=-=-=-=-=-=-=-=-=-=-

    function gameOver() {
      loss = loss + 1;
      document.getElementById("los").innerHTML = loss;
      alert('You Lose!');
      document.getElementById("btn").disabled = false;
      document.getElementById("m").removeEventListener('click', gameOver);
      theLeftSide.lastChild.removeEventListener('click', nextLevel)
      while (theLeftSide.firstChild) {
          theLeftSide.removeChild(theLeftSide.firstChild);
      }
      theRightSide.innerHTML = '';
       numberOfFaces = document.getElementById('numFace').value; //5;
       newStart = true;
    }
=-=-=-=-=-=-=-=-=-=-=-=-=-=-
Your collection so far: <span id="smiles">0</span> Smiles<br>
document.getElementById('smiles').innerHTML = numberOfFaces; //nextlevel
document.getElementById('smiles').innerHTML = 0;             //game over
```

Add a HINT

```
<button id='hint' onclick="giveHint()">Hint?</button>

<span id="warn">It will cost ya...</span>


function giveHint() {
    let hintImg = theLeftSide.lastElementChild;
    hintImg.style.width = '105px';
    hintImg.style.height = 'auto';
}
```

It will cost you!

```
numberOfFaces = numberOfFaces -1
document.getElementById('smiles').innerHTML = numberOfFaces;

function giveHint(){
    let hintImg = theLeftSide.lastChild;
    hintImg.style.width = '105px';
    hintImg.style.height = 'auto';
    numberOfFaces = numberOfFaces -1;
    document.getElementById('smiles').innerHTML = numberOfFaces;
    setTimeout(
        function(){
            hintImg.style.width = '100px';
            hintImg.style.height = 'auto';
        }
        ,3000
    );
}
```

You can also let the player know how many smiles they have collected…

Final Code (next page)

Center the Main section – Add this at the top of your script

```
    <script>
        let x = window.innerWidth;
        let y = x - 1000;
        if(y>0){
            document.getElementById('board').style.position = "absolute";
            document.getElementById('board').style.left = (y/2) + "px";
        }
```

Turn the number input into a range input

```
<input id="numFace" type="number" value="1" style="width: 40px">

<label for="xFaces"> How many faces do you want to generate: </label>
1
<input type="range" name="xFaces" id="xFaces" min="1" max="10" value="5" onchange="getFaces()" >
10 | your selection:
<span id="numFace" name="numFaces" >5</span>

function getFaces(){
    document.getElementById('numFace').innerHTML = document.getElementById('xFaces').value;
}
```

Also… change *.value* to *.innerHTML* in 3 places using Edit > Replace (or do it manually on lines 61, 62, 108)

Handling Errors…

```
function giveHint(){
    try {
        let hintImg = theLeftSide.lastChild;
        hintImg.style.width = '105px';
        hintImg.style.height = 'auto';
        numberOfFaces = numberOfFaces -1;
        document.getElementById('smiles').innerHTML = numberOfFaces;
        setTimeout(
            function(){
                hintImg.style.width = '100px';
                hintImg.style.height = 'auto';
            }
            ,5000
        );
    }
    catch(err) {
        alert("Hey… ya gotta start the game before asking for a hint")
    }
}
```

It will cost you show / hide

```
    <button id="hint" onclick="giveHint()" onmouseover="warning()">Hint?</button>
    <span id="warn" style="display: none;">It will cost ya...</span>

function warning(){
    document.getElementById('warn').style.display = 'inline-block';
    setTimeout(
        function(){
            document.getElementById('warn').style.display = 'none';
        }
        ,2000
    );
}
```

Try the timer with an arrow function…

```
setTimeout(()=>{document.getElementById('warn').style.display = 'none';},2000);
```

Base Code

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
        content="width=device-width, initial-scale=1.0">
    <title>Matching Game</title>
    <style>
        img {position: absolute;}
        div {
            position: absolute;
            width: 500px;
            height: 500px;
        }
        #rightSide {
            left: 500px;
            border-left: 1px solid;
        }
    </style>
</head>
<body>
  <h1>Matching Game</h1>
  <button onclick="generateFaces()">Start Game</button>
  <p>Click on the extra smiling face on the left.</p>
  <main id="board">
      <div id="leftSide"><img style="top: 100px; left: 100px;"></div>

      <div id="rightSide"></div>
  </main>
  <script>
    let numberOfFaces = 5;
    const theLeftSide = document.getElementById("leftSide");
    const theRightSide = document.getElementById('rightSide');

    // theLeftSide var now controls all properties/methods of the div element with the id “leftSide”
    function generateFaces() {
        for (i = 1; i <= numberOfFaces; i++) {  // standard for loop
            let randomTop = Math.floor(Math.random() * 400);
            let randomLeft = Math.floor(Math.random() * 400);
            const face = document.createElement("img");
                face.src = "../images/smile.png";
                face.style.top = randomTop + "px";
                face.style.left = randomLeft + "px";
            theLeftSide.appendChild(face);
        }
        const leftSideImages = theLeftSide.cloneNode(true);
        leftSideImages.removeChild(leftSideImages.lastChild);
        theRightSide.appendChild(leftSideImages);
        theLeftSide.lastChild.addEventListener('click', nextLevel);
        document.getElementById("board").addEventListener('click', gameOver);

    }
```

```
    function nextLevel(event) {
        event.stopPropagation();
        while (theLeftSide.firstChild) {
            theLeftSide.removeChild(theLeftSide.firstChild);
        }
        theRightSide.innerHTML = ''; //alturnative to remove child loop
        numberOfFaces += 5;
        generateFaces();
    } //End of nextLevel function

    function gameOver() {
        alert('You Lose!');
        document.getElementById("m").removeEventListener('click', gameOver);
        theLeftSide.lastChild.removeEventListener('click', nextLevel)
        while (theLeftSide.firstChild) {
            theLeftSide.removeChild(theLeftSide.firstChild);
        }
        theRightSide.innerHTML = '';
        numberOfFaces = 5;
    }

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

Bonus Code

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
        content="width=device-width, initial-scale=1.0">
    <title>Matching Game</title>
    <style>
        img {position: absolute;}
        div {
            position: absolute;
            width: 500px;
            height: 500px;
        }
        #rightSide {
            left: 500px;
            border-left: 1px solid;
        }
    </style>
</head>
<body>
  <h1>Matching Game</h1>
  <button onclick="generateFaces()" id="btn">Start Game</button>
  <p>Click on the extra smiling face on the left.<br>
      Wins = <span id="win">0</span> Loses = <span id="los">0</span>
      <br>How many faces do you want to generate:
      <input id="numFace" type="number" value="1" style="width: 40px">
  </p>


  <main id=" board">
      <div id="leftSide"><img style="top: 100px; left: 100px;"></div>

      <div id="rightSide"></div>
  </main>
  <script>
    let wins = 0; //Global Variable
    let loss = 0; //Global Variable
    let numberOfFaces; // = 5;
    let numFace;  // static
    let newStart = true; // check for new game
    const theLeftSide = document.getElementById("leftSide");
    const theRightSide = document.getElementById('rightSide');

    // theLeftSide var now controls all properties/methods of the div element with the id “leftSide”
```

```
function generateFaces() {
    if(newStart == true){
      numberOfFaces = document.getElementById('numFace').value;
      numFace = document.getElementById('numFace').value;
    }
    document.getElementById("btn").disabled = true;
    for (i = 1; i <= numberOfFaces; i++) {  // standard for loop
        let randomTop = Math.floor(Math.random() * 400);
        let randomLeft = Math.floor(Math.random() * 400);
        const face = document.createElement("img");
            face.src = "../images/smile.png";
            face.style.top = randomTop + "px";
            face.style.left = randomLeft + "px";
        theLeftSide.appendChild(face);
    }
    const leftSideImages = theLeftSide.cloneNode(true);
    leftSideImages.removeChild(leftSideImages.lastChild);
    theRightSide.appendChild(leftSideImages);
    theLeftSide.lastChild.addEventListener('click', nextLevel);
    document.getElementById("board").addEventListener('click', gameOver);

}
function nextLevel(event) {
    wins = wins + 1;
    document.getElementById("win").innerHTML = wins;
    newStart = false;
    event.stopPropagation();
    while (theLeftSide.firstChild) {
        theLeftSide.removeChild(theLeftSide.firstChild);
    }
    theRightSide.innerHTML = ''; //alturnative to remove child loop
    numberOfFaces = Number(numberOfFaces) + Number(numFace);  //5;
    generateFaces();
} //End of nextLevel function

function gameOver() {
    loss = loss + 1;
    document.getElementById("los").innerHTML = loss;
    alert('You Lose!');
    document.getElementById("btn").disabled = false;
    document.getElementById("m").removeEventListener('click', gameOver);
    theLeftSide.lastChild.removeEventListener('click', nextLevel)
    while (theLeftSide.firstChild) {
        theLeftSide.removeChild(theLeftSide.firstChild);
    }
    theRightSide.innerHTML = '';
```

```
        numberOfFaces = document.getElementById('numFace').value;  //5;
        newStart = true;
    }

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