Assignment: Matching Game

You submitted!

Your work is ready to be reviewed by classmates. Next, you need to review your classmates' work. We'll email you when your grade is ready. Your grade should be

Review a Classmates' Work (/learn/html-css-javascript/peer/dwdxh/matching-game/give-feedback)

Help Center

ready
by
November
18,
11:59
PM
PT.

Instructions (/learn/html-css-javascript/peer/dwdxh/matching-game)

My submission (/learn/html-css-javascript/peer/dwdxh/matching-game/submit)

Discussions (/learn/html-css-javascript/peer/dwdxh/matching-game/discussions)

Instructions

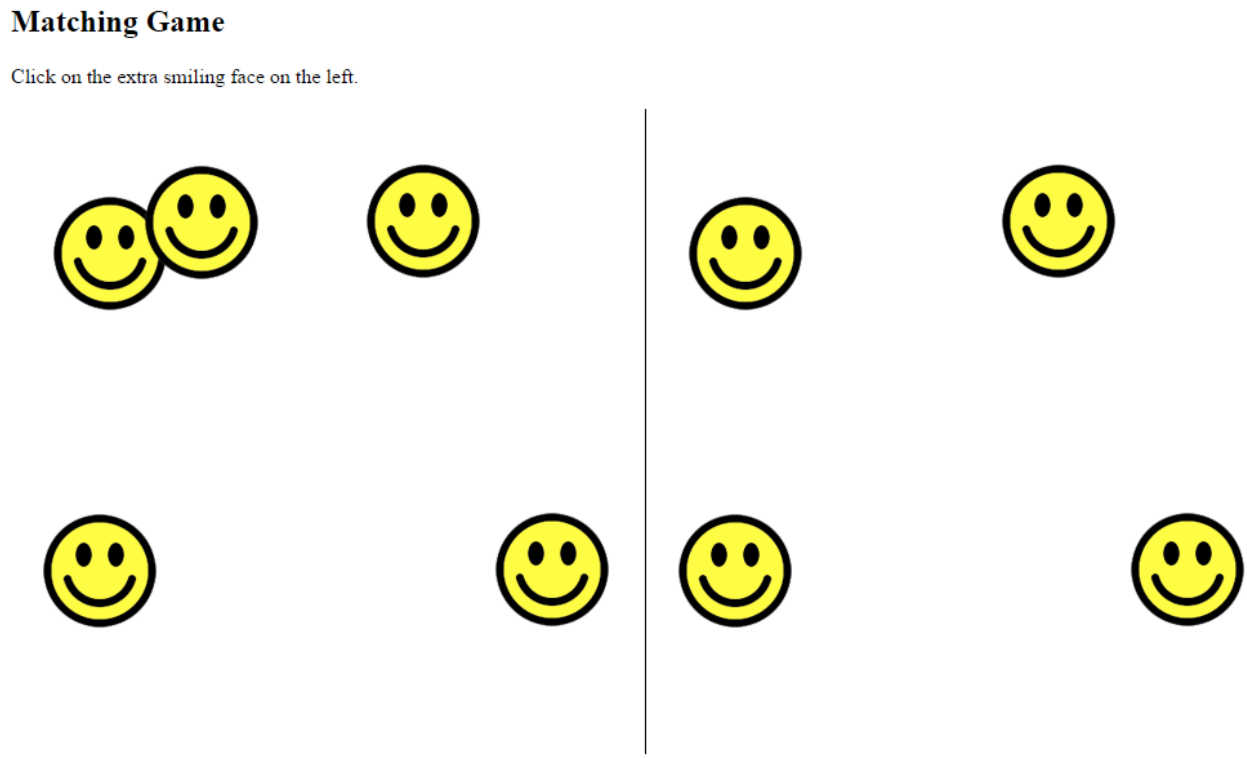
By making a game, you will gain experience in JavaScript as well as DOM handling and some CSS.

Instructions **less**

The Task

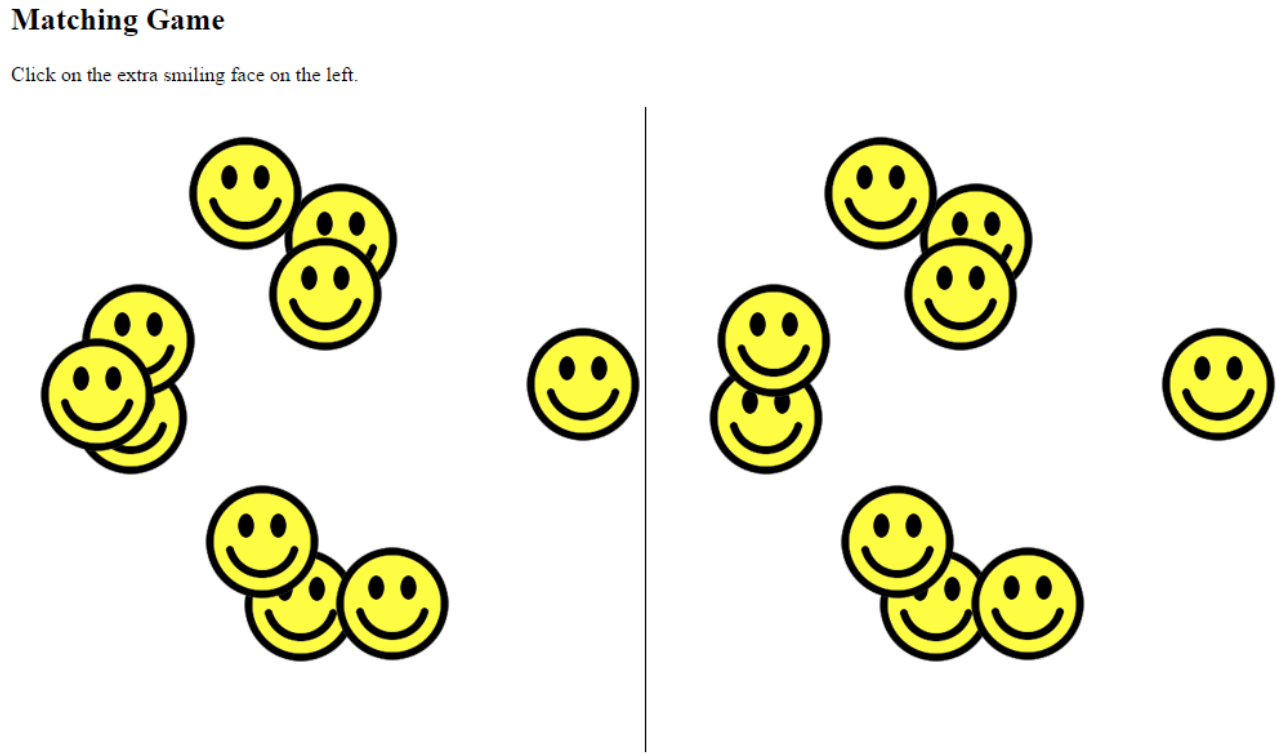
This assessment task requires you to make an interactive game. Please see the accompanying video for an example walk-through. Some of the skills in this assessment task are demonstrated in the Example DOM Project, and the code for that project has been released in the system. You are recommended to look at that project and understand it prior to starting this assessment task.

When the game starts, five 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 deleted 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 extra face shown on the left.

For example, let's imagine you are playing the game shown in the previous figure. After clicking on the extra face (at the top) all the faces disappear and the following new set of faces are shown. 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.

Technical Overview

The text instructions are handled by simple HTML.

Two *div* elements are included in the *body*, like this:

```
<div id="leftSide"></div>

<div id="rightSide"></div>
```

The first div is used to store all the faces shown on the left side. The second div is used to store all the faces shown on the right side.

The line shown in the middle of the web page is created by applying a style rule which tells the browser to show a border line for only the left side of the *rightSide* div.

The faces are dynamically created by JavaScript. First, all the faces are generated on the left side, under the div with id 'leftSide'. Then *cloneNode(true)* is used to clone all the faces to the div with id 'rightSide', so there is an exact copy. The last child in this new branch is then deleted, so that when the user looks at the screen there is one extra face on the left side compared to the right side.

The event handling is also applied by JavaScript. It will help you to understand the *Adding Events Using JavaScript* lesson and accompanying examples in the course. There are two event handlers:

1. One *onclick* event handler is applied to the extra face on the left side which the user needs to click on. When the event is triggered a variable containing the number of faces to be generated is increased by 5, and then the process of generating and displaying the faces begins again.

2. The second *onclick* event handler is applied to the body. If this function is triggered, it means the player has failed to select the correct face and the game is over. When this event is triggered an appropriate message is shown and the two event handler functions are removed.

The smiling face is a simple image called *smile.png*, which is given to you.

http://home.cse.ust.hk/~rossiter/mooc/matching_game/smile.png (http://home.cse.ust.hk/~rossiter/mooc/matching_game/smile.png)

Which Browser to Use

This project was developed using the Chrome browser. The project has been checked on the latest versions of Firefox, Internet Explorer, Safari, and Opera and works well in all of them. However, to avoid any potential trouble with inconsistencies between browsers it may be wise for you to use Chrome.

Discussion Video

Please watch the accompanying video, which shows and discusses the completed task.

What to Submit

You can only submit one single HTML file, which includes the JavaScript code within the file. That means there are no links to external JavaScript files or CSS files in this assessment task. The image file *smile.png* is given to you on the web site.

What to Submit

There are 4 parts of this assessment. You need to submit 1 file for each part. For each part you can only submit one single HTML file, which (for parts 2, 3 and 4) includes JavaScript code within it. That means there are no links to external JavaScript files or CSS files in this assessment task.

Here are the more specific requirements and further information for both parts.

Overview of the Parts

Part 1 – developing the web page content without JavaScript

Part 2 – generating the left side images

Part 3 – handling the right side

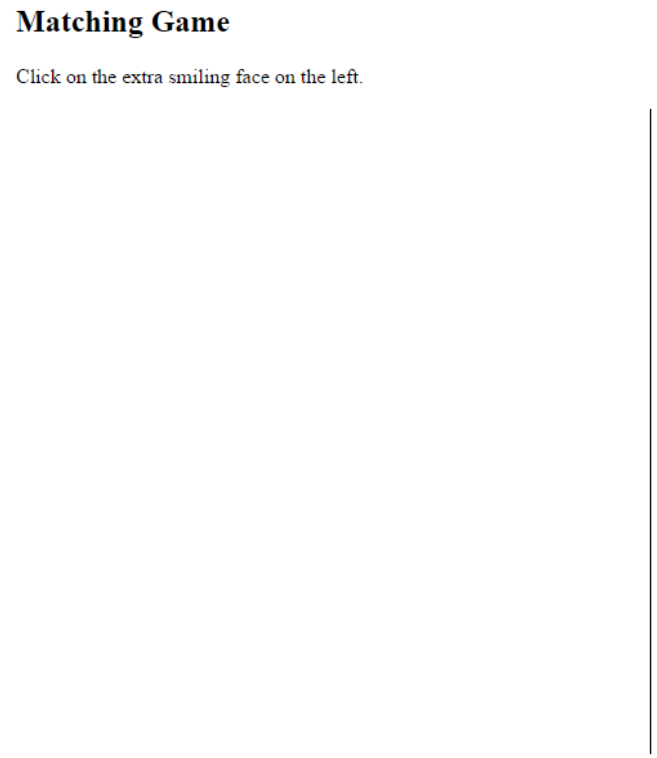
Part 4 – finishing the game

Part 1 Requirements.

For Part 1: [Web page content without JavaScript]

- The web page, without JavaScript is created.

The result will look like this when viewed in a browser. As you can see, only the instructions and the middle line are visible.



For part 1, the content of the file is as follows.

Instruction text

- A title e.g. 'Matching Game'. This could be any appropriate element such as *h1*, *h2*, or *h3*
- A message containing the instructions e.g. 'Click on the extra smiling face on the left.' This could be any appropriate element such as *p*.

Two divs

- `<div id="leftSide"></div>`

- this will contain all the faces shown on the left side

- `<div id="rightSide"></div>`

- this will contain all the faces shown on the right side

Style

- At least three style rules are needed
- `position:absolute` needs to be applied to all `img` . This is so that we can fix the exact position of any image later. Although we haven't actually added any images to the web page at this stage, we are adding this style rule now so that we can easily control their exact position later.
- `position:absolute` needs to be applied to all `div` . (However, you will probably find that you do not have to set the *leftSide* div position. Only the rightSide div position needs to be set, as discussed below). The two divs should have the same width and height e.g. `width:500px; height:500px`
- We will use a clever trick with style to show the line in the middle of the screen. We will

achieve this by adding a border for the left side of the second div (the *rightSide* div). Because we want to apply this to only one of the divs we cannot include the rule in the previously mentioned rule concerning divs. Instead, we will make a rule which will be applied to only one of the divs e.g.

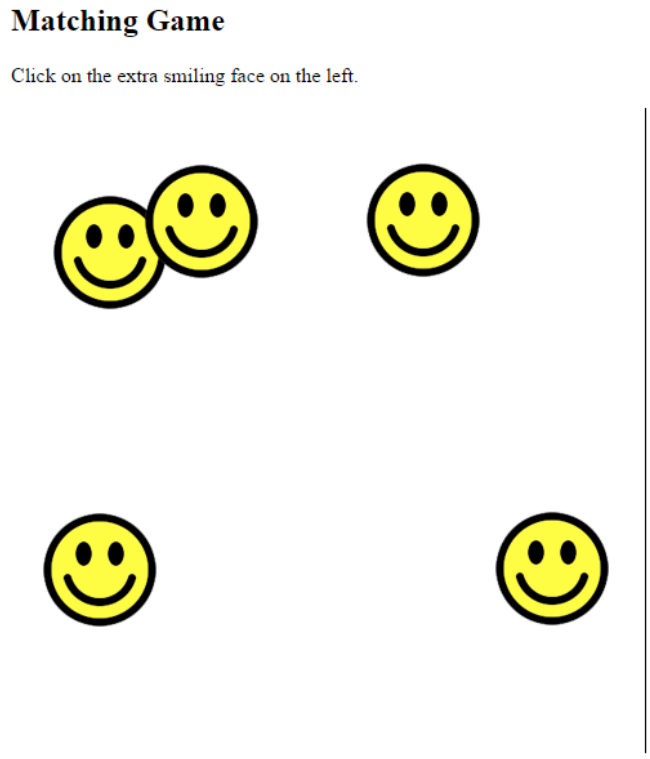
```
#rightSide { left: 500px;
             border-left: 1px solid black }
```

In this way the rule is automatically applied to the element which has the id *rightSide*, so we give this id to the div which contains the right side images. In addition to handling the border, the *rightSide* div is also positioned at the appropriate place using this rule.

Part 2 Requirements.

For Part 2: [Generating the left side images]

- For this stage you need to add JavaScript code which generates *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.



The JavaScript code for this stage is executed when the page is loaded. It does the following.

Variables

- A variable called *numberOfFaces* is created and set to the number 5.
- A variable called *theLeftSide* is created. This points to the left side div e.g.

```
var theLeftSide =
    document.getElementById("leftSide");
```

A function *generateFaces()* is created

- In this function the faces are created in a loop. The loop executes *numberOfFaces* times. In each iteration:

  - An *img* is created using *createElement()*

  - The img *src* attribute is set so that the appropriate image filename is used e.g. `img.src="smile.png";`

  - The position of an image is controlled by the *top* and *left* values. The img *top* value is set to be a random value in an appropriate range. For example, if the height of the div is 500 and the height of the image is 100, then an appropriate range for the random *top* value of the newly created image is 0 to 400. Use *style.top* to set this.

  - The img *left* position is set to be a random value in an appropriate range. For example, if the width of the div is 500 and the width of the image is 100, then an appropriate range for the random *left* value of the newly created image is 0 to 400. Use *style.left* to set this.

  - Add the newly created image to the *leftSide* div using *appendChild()*

Start the Function

- When the page is loaded, call the function *generateFaces()* to run it

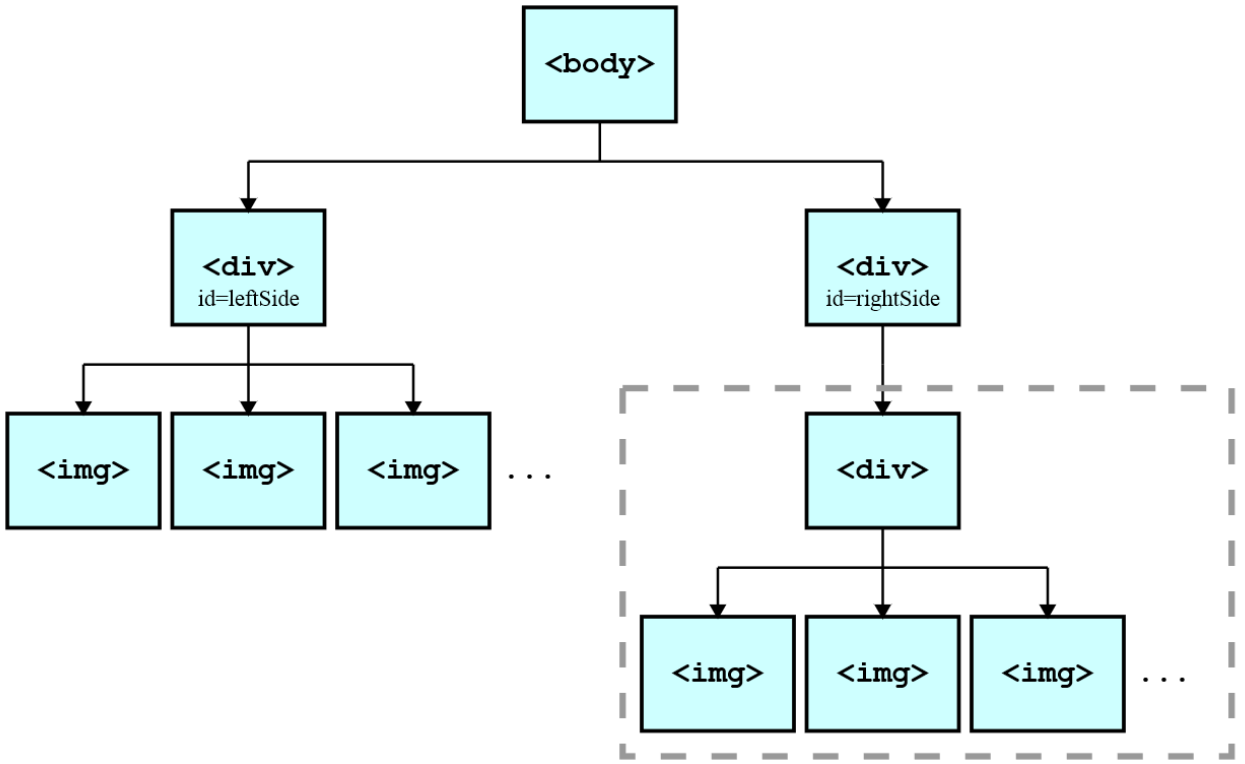
Part 3 Requirements.
For Part 3: [Handling the right side]

For Part 3 you need to extend the JavaScript code developed in part 2 to handle the right side.

- A new variable called *theRightSide* is created. This points to the right side div e.g.

```
var theRightSide =
    document.getElementById("rightSide");
```

- After all the images have been created and added to the *leftSide* div, use *cloneNode(true)* to copy the leftSide div e.g. leftSideImages = theLeftSide.cloneNode(true);
- Then, delete the last child of *leftSideImages*
- Then, add *leftSideImages* to the *rightSide* div
- The DOM will look like this:



Copy of leftSide with the last child removed

- It is totally OK to have a div under a div. In fact, this is quite common.

Part 4 Requirements
For Part 4: [Finishing the game]

For Part 4 you need to extend the JavaScript code developed in parts 2 and 3, mainly to handle the events and game logic.

Variables

- A variable called *theBody* is created. This points to the body e.g.

```
var theBody =
 document.getElementsByTagName("body")[0];
```

Add an event handler function to the extra face

- After cloning the node, you need to add an *onclick* event handler function to the last child of the left side, which is the face that the player is supposed to click on
- Your code for constructing the event handler function for that node will look similar to this:

```
theLeftSide.lastChild.onclick=
    function nextLevel(event){

        event.stopPropagation();
```

```
        numberOfFaces += 5;

        generateFaces();


};
```

- The code `theLeftSide.lastChild.onclick=`

```
        function nextLevel(event){ ...}
```

means that we are constructing a function which will be executed when the user clicks on the last child node in *theLeftSide*. The name of the function being created (`nextLevel()`) is not particularly important, it can be anything appropriate.

- The line `event.stopPropagation();` 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.
- The 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

- The line `generateFaces();`

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.

Add an event handler function to the body

- You need to add another function for handling the situation when the player clicks on anything except the correct face
- Your code for constructing the event handler function will look similar to this:

```
theBody.onclick = function gameOver() {

    alert("Game Over!");


    theBody.onclick = null;

    theLeftSide.lastChild.onclick = null;


};
```

- The line

```
theBody.onclick = function gameOver() {
```

means that we are constructing a function which will be executed when the user clicks on *theBody*. The name of the function being created (`gameOver()`) is not particularly important, it can be anything appropriate.

- The line `alert("Game Over!");`

is one way to show a message to the user

- The line `theBody.onclick = null;`

means that from now onwards nothing will happen when the user clicks anywhere in the web page

- The line

```
theLeftSide.lastChild.onclick = null;
```

means that from now onwards nothing will happen when the user clicks on the extra face.

Delete the child nodes

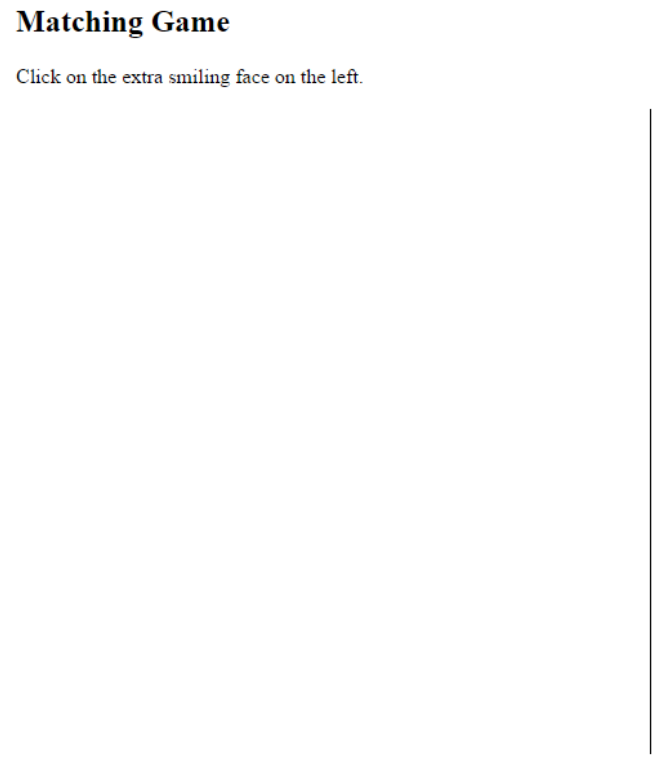
- Remember that each time the player clicks on the correct face all faces are removed and a new set of faces are generated. So that means at the appropriate place all children under the *leftSide* div and *rightSide* div need to be deleted. You previously learnt how to delete all child nodes on the course using a while loop.

Review Criteria **less**

For Part 1: [Web page content without JavaScript]

- The web page, without JavaScript is created.

The result will look like this when viewed in a browser. As you can see, only the instructions and the middle line are visible.



For part 1, the content of the file is as follows.

Instruction text

- A title e.g. 'Matching Game'. This could be any appropriate element such as *h1*, *h2*, or *h3*
- A message containing the instructions e.g. 'Click on the extra smiling face on the left.' This could be any appropriate element such as *p*.

Two divs

-

- this will contain all the faces shown on the left side

-

- this will contain all the faces shown on the right side

Style

- At least three style rules are needed
- position:absolute needs to be applied to all img. This is so that we can fix the exact position of any image later. Although we haven't actually added any images to the web page at this stage, we are adding this style rule now so that we can easily control their exact position later.
- position:absolute needs to be applied to all div. (However, you will probably find that you do not have to set the *leftSide* div position. Only the rightSide div position needs to be set, as discussed below). The two divs should have the same width and height e.g. width:500px; height:500px
- We will use a clever trick with style to show the line in the middle of the screen. We will

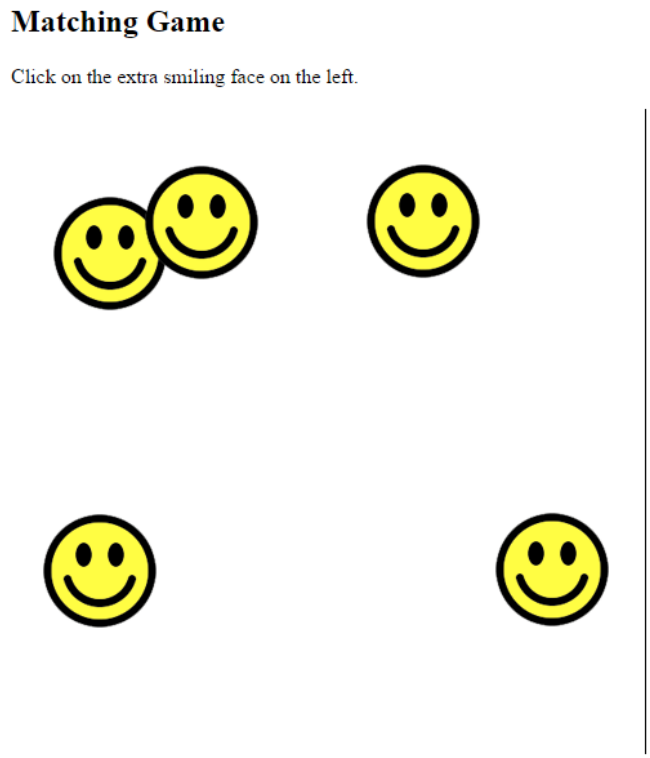
achieve this by adding a border for the left side of the second div (the *rightSide* div). Because we want to apply this to only one of the divs we cannot include the rule in the previously mentioned rule concerning divs. Instead, we will make a rule which will be applied to only one of the divs e.g.

```
#rightSide { left: 500px;
        border-left: 1px solid black }
```

In this way the rule is automatically applied to the element which has the id *rightSide*, so we give this id to the div which contains the right side images. In addition to handling the border, the *rightSide* div is also positioned at the appropriate place using this rule.

For Part 2: [Generating the left side images]

- For this stage you need to add JavaScript code which generates *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.



The JavaScript code for this stage is executed when the page is loaded. It does the following.

Variables

- A variable called *numberOfFaces* is created and set to the number 5.
- A variable called *theLeftSide* is created. This points to the left side div e.g.

```
var theLeftSide =
   document.getElementById("leftSide");
```

A function *generateFaces()* is created

- In this function the faces are created in a loop. The loop executes *numberOfFaces* times. In each iteration:

- An *img* is created using *createElement()*

- The img *src* attribute is set so that the appropriate image filename is used e.g. img.src="smile.png";

- The position of an image is controlled by the *top* and *left* values. The img *top* value is set to be a random value in an appropriate range. For example, if the height of the div is 500 and the height of the image is 100, then an appropriate range for the random *top* value of the newly created image is 0 to 400. Use *style.top* to set this.

- The img *left* position is set to be a random value in an appropriate range. For example, if the width of the div is 500 and the width of the image is 100, then an appropriate range for the random *left* value of the newly created image is 0 to 400. Use *style.left* to set this.

- Add the newly created image to the *leftSide* div using *appendChild()*

Start the Function

- When the page is loaded, call the function *generateFaces()* to run it

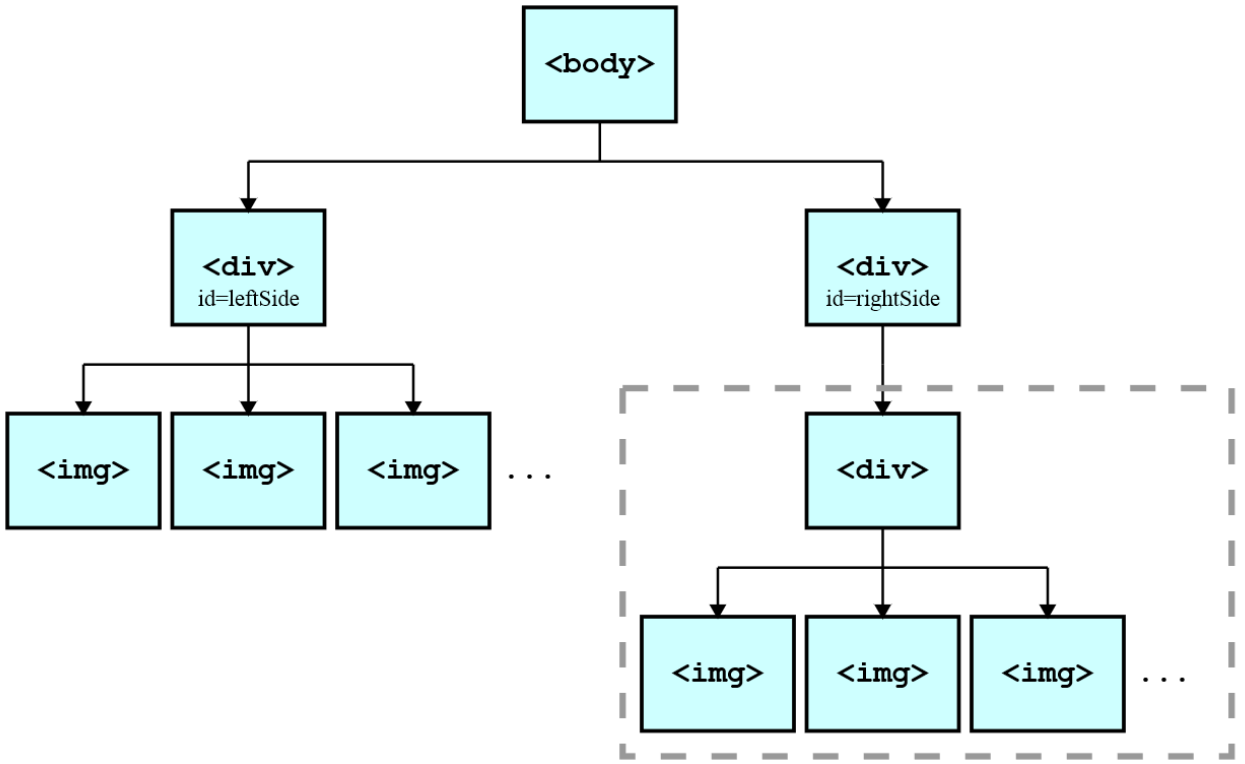
For Part 3: [Handling the right side]

For Part 3 you need to extend the JavaScript code developed in part 2 to handle the right side.

- A new variable called *theRightSide* is created. This points to the right side div e.g.

```
var theRightSide =
  document.getElementById("rightSide");
```

- After all the images have been created and added to the *leftSide* div, use *cloneNode(true)* to copy the leftSide div e.g. leftSideImages = theLeftSide.cloneNode(true);
- Then, delete the last child of *leftSideImages*
- Then, add *leftSideImages* to the *rightSide* div
- The DOM will look like this:



Copy of leftSide with the last child removed

- It is totally OK to have a div under a div. In fact, this is quite common.

For Part 4: [Finishing the game]

For Part 4 you need to extend the JavaScript code developed in parts 2 and 3, mainly to handle the events and game logic.

Variables

- A variable called *theBody* is created. This points to the body e.g.

```
var theBody =
 document.getElementsByTagName("body")[0];
```

Add an event handler function to the extra face

- After cloning the node, you need to add an *onclick* event handler function to the last child of the left side, which is the face that the player is supposed to click on
- Your code for constructing the event handler function for that node will look similar to this:

```
theLeftSide.lastChild.onclick=
   function nextLevel(event){

      event.stopPropagation();

      numberOfFaces += 5;
```

```
    generateFaces();
};
```

- The code theLeftSide.lastChild.onclick=

```
    function nextLevel(event){ ...}
```

means that we are constructing a function which will be executed when the user clicks on the last child node in *theLeftSide*. The name of the function being created (nextLevel()) is not particularly important, it can be anything appropriate.

- The line event.stopPropagation(); 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.
- The 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

- The line generateFaces();

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.

Add an event handler function to the body

- You need to add another function for handling the situation when the player clicks on anything except the correct face
- Your code for constructing the event handler function will look similar to this:

```
theBody.onclick = function gameOver() {
  alert("Game Over!");

  theBody.onclick = null;
  theLeftSide.lastChild.onclick = null;
};
```

- The line

```
theBody.onclick = function gameOver() {
```

means that we are constructing a function which will be executed when the user clicks on *theBody*. The name of the function being created (gameOver()) is not particularly important, it can be anything appropriate.

- The line alert("Game Over!");

is one way to show a message to the user

- The line theBody.onclick = null;

means that from now onwards nothing will happen when the user clicks anywhere in the web page

- The line

theLeftSide.lastChild.onclick = null;
means that from now onwards nothing will happen when the user clicks on the extra face.

Delete the child nodes

- Remember that each time the player clicks on the correct face all faces are removed and a new set of faces are generated. So that means at the appropriate place all children under the *leftSide* div and *rightSide* div need to be deleted. You previously learnt how to delete all child nodes on the course using a while loop.

‹ (/learn/html-css-javascript/supplement/xaomc/faq-for-advanced-javascript-assessment-task)

› (/learn/html-css-javascript/lecture/gCrVW/concluding-message)

‹ (/learn/html-css-javascript/supplement/xaomc/faq-for-advanced-javascript-assessment-task)

› (/learn/html-css-javascript/lecture/gCrVW/concluding-message)