Simulating Ball Movement

Identifying Web Page elements

Elements on a web page are identified using an "id". Every HTML element you define needs to have a unique identifier.

For example, the following header HTML element has an id of "title":

```
<h1 id="title">Welcome</h1>
```

In JavaScript, you have the ability to identify an element based on its id, and change any attributes that are associated with the selected element (for example, styles). This is made possible thanks to your browser's interpretation of the HTML code you have written. When your HTML page is loaded in a browser, the browser creates a tree structure representation of all the HTML elements you described in your page. This structure makes it easy for web programming languages like JavaScript to find elements on the page using their respective id and other attributes.

In other words, when you create an HTML page with the following code:

```
<html>
<body>
    <h1 id="title">Welcome</h1>
</body>
</html>
```

The browser loading up your page will see this instead:

```
HTML
|— HEAD
|— BODY
    └── #text:
    └── H1 id="title"
        └── #text: Welcome
```

Notice how the elements are nested and structured in a way that would make it easy for JavaScript to find each one of them using the id.

One more thing: Adding styles to HTML elements:

You can define how HTML elements on a web page should look like using CSS. This is a set of rules that describes the style of every HTML element you create (for example, font, color, etc).

CSS styles can be declared inside an HTML tag using the "style" attribute. For example, the following HTML code will result in the text "Welcome" being displayed in red:

```
<h1 id="title" style="color: red">Welcome</h1>
```

Your task in this activity is to do the following:

- Using HTML and CSS, display a ball on the page. The div element you create needs to have "ball" as the id. This code should be added to the `index` file.

The "ball" element should have the following CSS attributes:

- z-index: 5;
- position: absolute;
- left: 0px;
- top: 200px;
- width: 100px;
- height: 100px;
- border-radius: 50%;
- background: black;

In the `ball.js` file, create a global variable called `ball` and set it to the HTML element `<div>..</div>` with id "ball" using the `getElementById()` built-in function.

Task

Display a ball on the web page and create a variable containing that element. The ball element should have an id of `ball`.

Adding Animation

In the last step, you created and displayed a ball on the Html page. Then, you added JavaScript code to capture the ball element in a JavaScript variable.

In this step, you're going to use that variable to change the position of the ball element on the page dynamically. You can achieve this by updating the style properties that control the element position.

Your task is to make the ball continuously move from the left of the page to the right.

To complete this task, you need to do the following:

- Declare a global variable called `velocity` and set its value to `100`. This variable controls the speed at which the ball moves across the page.
- Declare another global variable called `positionX`. This variable represents the position of the ball on the x-axis. The value of this variable should initially be `0`.
- Inside the `moveBall()` function, add logic to update the `positionX` variable by adding the value of `velocity` to it. Then set the ball's position to this new value.

After making these changes, the ball should move from left to right continuously.

Hint:

- *You can access the elements position using ball.style object.*
- *Make sure you append the string `px` to the end of the new position value.*

Task

- Add animation to move the ball across the page.

Adding Screen Edge Detection on the x-axis

In the last step, you added animation to allow the ball to move across the page.

Now, try adding edge detection so that the ball can bounce between two specific areas on the x-axis on the page.

Your task is to write an HTML and JavaScript code that will allow the ball to move between two fixed positions on the x-axis on the page.

To accomplish this task, you need to do the following:

- Define a global variable called `reverse` and set its value to `false`. This variable is a boolean (can either be `true` or `false`) and it will help you define which direction the ball should move in.
  - The `reverse` variable will change value every time the ball hits the edge of a defined area on the screen.
  - The area is defined by the variables `Xmin` and `Xmax`.
- Inside the `moveBall()` function, add logic to set the ball position on the x-axis based on the value of `reverse`:
  - When `reverse` is `false`, the ball moves from left to right (ball position + velocity.)
  - When `reverse` is `true`, the ball moves from left to right (ball position - velocity.)
- After that, inside the `moveBall()` function, add logic to compare the variable `positionX`, `Xmin` and `Xmax`. if the ball position (`positionX`) is greater than `Xmax` or equals `Xmin`, then the value of `reverse` should be switched.

Adding these changes will make the ball move between two points on the x-axis across the screen.

Hint: you can use the `!` operator to switch the variable of boolean.

Task

- Write code that will make the ball move between two set positions on the x-axis on the page.

Adding Screen Edge Detection on the y-axis

Now that you have added edge detection and the ball can bounce between two specific points on the x-axis on the page, it is time to increase the complexity.

Now add edge detection so that the ball can bounce between two specific areas on both the x-axis and y-axis on the page.

Your task is to add JavaScript code that will allow the ball to move between two fixed positions on the x-axis and y-axis on the page.

To accomplish this task, you need to do the following:

- Define a global variable called `positionY` and set its value to `0`.
- Inside the `moveBall()` function, note the two variables called `Ymin` and `Ymax`. These variables define the two points on the y-axis of the page where the ball should bounce.
- Update the logic you created in step 3 to include checks for the y-axis position of the ball.

After implementing these changes, the ball should bounce diagonally between the defined points on both the x-axis and y-axis.

Task

- Add an HTML and a JavaScript code that will allow the ball to move between two fixed positions on the x-axis and y-axis on the page.