Scoring and Determining Proximity of 2 Elements:

Contents

Detecting location of two elements: Intro

What if we want to detect whether two elements are located in the same place in a document?

For our game, we want to know whether the hero and the bad thing are in the same place because it indicates that the hero got hit by the bad thing and thus loses.

We also want to know when the hero is in the same place on the screen as a goodie because then the hero collects the goodie.

Detecting whether two elementsare in the same place has many, many uses beyond this game. One very important use is to determine if two elements are overlapping on a web page, thus blocking content readability. (I.e., if I am resizing or relocating elements based on user's preference when they are setting up a dynamic user interface, I'd want to know whether one object is on top of another). Another example – if you're doing a fund raiser or anything that involves reaching a goal, and you have a bar or some object that moves based on how close you are to the goal, you might want to have something detecting when the bar touches the goal marker to indicate that the goal has been reached.

Detecting whether 2 elements are in the same place:

To start, we will detect whether our hero and the bad thing are in the same place.

There are variables that hold the number of pixels down from the top and the number of pixels over from the left for both the hero and the bad thing.

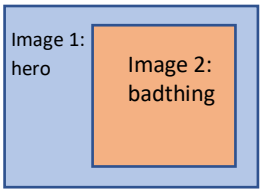
Luckily, we made both of these variables global, which means that more than one function can access those variables.

So you could just check to make sure that the hero's top is equal to the bad thing's top and the hero's left is equal to the bad thing's left.

```
if ((htop === btop) && (hleft === bleft)){
        …
}
```

Problem: pixels are small. In order for the hero's top and left pixel is in the exact same place as the bad thing's top and left pixel, people would have to move them to EXACTLY the same place on your screen, with no leeway!

We, as humans, usually aren't that precise. We want to say that the hero and bad thing are on top of each other if they're both within a range of pixels. To give an example, below we have an image of a hero and an image of a bad thing. I would say that they are in the same place. But their top left pixels aren't in the exact same place.



In this case, even though hleft and htop and bleft and btop aren't exactly the same, I'd consider the hero and the bad thing to be in the same place.

In order to allow the two images to be considered to be in the same place, you will want to check if bleft is close to hleft, and, if so, then it would be considered on top of each other. If you want to see if the two left corners are within 40 pixels of each other (you may want to make this bigger…), you'd do the following:

```
if ((hleft> bleft-20) && (bleft < cleft + 20)) {
                    …
}
```

Now when hleft is within 40 pisels of bleft (say hleft is 40, then cleft can be anywhere from 20 (40 – 20) to 60 (40 + 20) the two images will be considered to be on top of each other. Equally,

```
if ((htop> btop-20) && (htop < btop + 20)) {

        …

}
```

checks to see whether the two tops are within 20 pixels of each other (again, you may want to make this range larger – 20 pixels is still pretty small!

Combined: you'd get:

```
if ((btop > htop - 20) && (btop < htop + 20) && (bleft > hleft - 20) && (bleft < hleft + 20)) {

…
```

Let's Try It:

Step 0: make sure you've got your hero and your bad thing on your html page. Make sure that the hero can be moved by clicking on the different buttons (there's a better, less annoying way to move the hero, but for now go with this). Also, make sure you've got a start button that starts your bad thing moving automatically on your web page.

Step 0b: In your javaScript file, make sure both the hero and the bad thing are positioned automatically and use global variables and make sure the 2 variables for positioning the hero and the two variables for positioning the bad thing are at the very top of your javascript file, outside and above all functions.

Step 1: under the global variables, create a new function. The purpose of this function will be to determine whether your hero and your bad thing are positioned on top of each other, so name it appropriately.

Step 2: This function has no parameters (it's been a while since we've had a function that didn't have an input parameter, but this one doesn't need one.

Step 3: Inside the function, add the if condition that checks to see if the hero and the bad guy are within a certain range of each other (like I did above). If they are, add an alert that says, "YOU LOSE!! Hero destroyed by bad thing!" (fill in a better description based on your game).

Step 4: For this function, we want it to happen whenever the bad thing moves.

In general, whenever you're checking the position of two things, you want to add the check to the thing that moves more or faster. So in this case, the bad thing is moving more and faster than the hero. So I'm going to call the function right after I move the bad thing in my MoveBad function.

Note that this is a new way of calling a function. Inside the MoveBad function, right after you reposition the bad thing using document.getElementById.style.top = btop,on the next line, call the function you wrote in Step 1 by simply writing the name of the function with the two parentheses. So, for instance, if I had named the function thisfunc, I'd have the following in the MoveBad code:

```
…

document.getElementById(par).style.top = btop + "px"
thisfunc()
```

Step 5: Save your code. Load it into the browser and test it. Move your hero around to the center of the screen – once you've moved it a bit, you may have to wait a bit until the bad thing actually hits your hero for testing. The other possibility is to slow the bad thing's movement speed down (by putting a bigger number in the setTimeout call in your MoveBad function). That way you have more time to move your hero under the bad thing, but then you still have to wait for your bad thing to fall on your hero.

If this doesn't work:

- Check your parentheses in your if condition. It's easy to miss an opening or a closing parenthesis, and then the whole thing just doesn't work at all.
- Check to make sure you initialized your hero's position from the top and the left to be exactly what you set them to be in the html code when you styled the hero.
- Check to make sure the name of the function and the call to the function are spelled exactly the same way.

If you want to double-check your numbers, and watch them, you can actually print out the values in the variables by adding a paragraph to your html code, and then changing its innerHTML to hold the variables of the for the htop, hleft, btop,and bleft as follows:

Step 5b: In your html code, add a paragraph. Give it the id of "debug" or something else unique.

Step 5c: Now in the function you just wrote in Step 1, above the if condition, add a line:

```
document.getElementById('debug').innerHTML = "Hleft:"+hleft+" Bleft:"+bleft+" Htop:"+htop+" Btop:"+btop
```

Step 5d: Save and test your code. You should now see in the paragraph the positons from the top and the left of your hero and your bad thing as they move. It's kind of cool!

Your Turn:

- ### (10 pts) Get the above function working
  so that when the hero and the bad thing are on top of each other, you get an alert saying, "You Lose!..."

Scoring:

In our game, we want there to be a score, and we want the score to increase every time the hero "catches" a goodie, or the hero's position and the goodie's position are in the same place. So we're basically doing something very similar to what we did in steps 1-5 for the goodies, only instead of getting an alert that "You Lose" when the two elements are in the same place, we want to increase a score.

While for our purposes, the score is used to keep track of how many goodies have been captured in a game, a scoring mechanism like this is used in many applications – for instance, all games ever. Another example would be keeping track of how many someone got right in an online quiz, or perhaps keeping track of how often a person chose a particular style in some preference tracking quiz. There are many, many uses for a scoring mechanism.

So let's add a scoring mechanism to the game:

Step 6: In your html code, add a paragraph. This paragraph is where the score will be showing up. So position it appropriately (where you want it to show up) on your page. Give it an id that is unique.

Step 7: In your javascript, at the top of the file, add a global variable that will hold the score. Initially set the score to be 0 (because initially your hero has no points).

Step 8: Now add another function. This function will be responsible for detecting whether the hero is over a goodie. So give it an appropriate name.

Step 9: Inside the function, add an if condition similar to that added in Step 3. Only check to see if the hero's left and top are within range of the goodie's left and top.

Step 10: Inside the if condition, if the hero and goodie are within range, add 1 to the score (e.g., score = score + 1).

Step 11: Now in the alert, instead of saying "You Lose", change it to say "You caught a goodie!" + score to show the score.

Step 12: use document.getElementById with the id of the paragraph added in step 6 and change the paragraph's innerHTML to be the score

Step 13: Now, again, you want this function to be called from the thing that moves faster and more often. So with the bad thing and the hero, the bad thing moved faster and more often. But with the hero and the goodie, the goodie stays put while the hero moves toward the goodie (while trying to avoid the bad thing). So you want to put the call to the function created in step 8 inside the move function for the hero (MoveUp, MoveLeft, MoveDown, MoveRight). You want to put the call right after the document.getElementById updates the position of the hero.

Step 14: Save your code. Load it in a browser and test it. When the hero gets over a goodie, the score should go up by 1.

Note that for testing, you might want to slow down the movements of the goodie so tht the hero can actually get to it and get over it (Honestly, there really is an easier, quicker way to move the hero – we'll see that in a future tutorial)

Your Turn:

- (10 pts) Get the scoring working

Starting the game:

Currently, you've got one button that starts the bad thing moving on your web page, and another button that starts the good thing jumping around your page. So if you want to start having your hero gather goodies while simultaneously dodging the bad thing, you have to click both buttons to make them both start happening.

You probably want one start button. There are many ways to do this, but probably the most straightforward is to create yet another button on your web page and have it call a startUp function, and then have the startUp function call both the MoveBad function and the ChangePos function.

Step 15: Create a button in your html page. Position it somewhere for the start of the game so that it isn't over or under another element on your web page. Give the button a value of "start game" and have the button call the function StartGame when it is clicked on.

Step 16: In your javaScript file, create a function called StartGame. It does not need a parameter.

Step 17: Inside the StartGame function, call MoveBad('b1') and ChangePos('c1'), making sure to replace b1 with the id of the bad thing in your game and replace c1 with the id of the goodie in your game.

Step 18: Save both files. Load into the browser. Test by clicking on the start game button. Now both the bad thing and the good thing should start moving simultaneously.

Step 18b: remove the other two start buttons from your html code that you no longer need.

Your Turn:

- (4 pts): Get the Start Game button working

You now have all the basic components of the game! Now make it work with keystrokes!