JavaScript: Tutorial 5

In tutorial 1 you learned:

1. How to add javaScript to your html code
2. How to write a function in js
3. How to call a function in your html using onClick() and buttons
4. How to write things out using an alert box
5. How to click on an image in your html code to call a function in your javaScript
6. How to add a parameter to your function and how to call a function with a value for the parameter
7. How to create and use a variable
8. How to use prompt box to put a value into a variable
9. What an if condition is, what an else-if condition is, and what an else condition is

In tutorial 2 you learned:

1. Using document.write to write out new html code, including
    a. Writing new styles
2. Combining variables with strings using the + sign

In tutorial 3 you learned:

1. How to modify existing html using document.getElementById()
    a. Using document.getElementById to change the width and the height of an existing element
    b. Using document.getElementById to change the src and the alt of an existing element
    c. Sending the id of an html element into the function to be used by document.getElementById()
2. How to read information from existing html using document.getElementById()
    a. How to read and modify the width and height, and then put back the newly modified width and height using document.getElementById()
3. How to use document.getElementById() to modify the style of an existing element
4. What the innerHTML is and how to read and modify that using document.getElementById()

In tutorial 4 you learned:

1. More about exactly what innerHTML is
2. How to join to the end of a variable or string using +
3. The difference between using + with numbers and + with strings of characters
    a. Using parseInt to change a string of characters to a number
4. Modifying the length of a list using if and prompts
5. An intro to a while loop

You've learned a lot! In this tutorial you'll be learning some miscellaneous tools to make programs more fun, and also a bit about how to find errors in your code.

Random numbers

Every computer game in the world uses random numbers. Things have to happen randomly or it's not much of a game. Random numbers are used any time you want something to happen unexpectedly – for instance, you may want random pictures to show up in a display, or you may want random words of encouragement to pop up in an app. Another example is in generated synthetic voices. If you just use a pure sin wave (a smooth sound wave), the resulting sound has that sort of tinny, synthetic sound to it. To make something sound more like a real voice, you need to add some breathiness. To add breathiness to a smooth sound wave, you just generate small random numbers and add them to the wave.

These are just a few of many uses for random numbers.

JavaScript has a random number generator built in. It looks like this:

```
x = Math.floor(Math.random() * 10)
```

This generates a random number between 0 and *10 (not including 10)* and puts that random number into the variable x.

The number 10 in the above example is what specifies that the largest random number to be generated will be 9. You can always change this number. For instance, if you have the following random number generator:

```
x = Math.floor(Math.random() * 3)
```

The possible random numbers that x might hold would be 0, 1, or 2 (not 3 - the smallest number is 0, and the largest is always one less than the number after the * (which is multiplication in javaScript).

Now, what if you want a random number between 5 and 15? You'd do the following:

x = Math.floor(Math.random() * *10*) + 5

The first part generates a random number between 0 and 9, and then, whatever that random number is, 5 is then added to it. So, for instance, if the random number generated is 0, then after 5 is added, x will hold 5. If the random number generated is 4, then, after 5 is added, x will hold 9, and if the random number generated is 9, then x will hold 14.

Thus the smallest number the random number can generate, which is 0, plus 5, will make the smallest random number 5, and the largest number the random number can generate, which is 9, plus 5,will make 14, so the range is from 5 to 14, which is what we wanted.

You can just memorize how to use this, or you can read the following explanation:

x = Math.floor(Math.random() * *9*)

1. Math.random() generates a random number between 0 and 1 (e.g., 0.4).
2. This number is multiplied by the range (in this case 9):
    a. Math.random() * 9 gives us 3.6
3. Math.floor(3.6) rounds down to the nearest integer, or 3
    a. It chops off everything after the decimal point
4. Finally, you need a variable to put all this in.
    a. x = 3

Let's use random numbers:

Step 1: In your html file, create an image. Give the image an id. You can make the image be of anything at all – I made mine be a closed curtain (like in a magic trick, with something hiding behind it) but it really doesn't matter what it is.

Step 2: In your html file, create a paragraph. Give the paragraph an id. Again, I don't really care what it says now – a function will change its content shortly

Step 3:. Create a button. Make the value "play game" and make the onClick call a function PlayGame().

Now in your .js:

Step 4: Create a function PlayGame() Note that this function doesn't need a parameter.

Step 5: Inside the function, generate a random number between 0 and 2 (not including 2) and place the random number in a variable as follows:

```
x = Math.floor(Math.random() * 2)
```

Step 6: Add to the function an if condition such that if x holds 1 (the random number generated was 1), change the image in step 1's src to be that of a coin head and create a variable y that holds the word, 'heads' (fyi, the id of my image is 'cid'):

```
if (x === 0) {
        y = 'heads'
        document.getElementById('cid').src = "coinhead.jpg"
}
```

Step 7: create an else condition that sets y to tails, and changes the .src of the image to be that of a cointail

Step 8: Below the else condition (at the bottom of the function) use document.getElementById and the y variable to change the paragraph created in step 2 to "computer generated " + y

Step 9: Save and test. When you click on a button, either a coin head should show up and the paragraph "computer generated heads" should show up, or a coin tail should show up and the paragraph should say, "computer generated tails"

Let's make it a game:

Step 10: Back to the html: add another image next to the first image. Give it a different id. I put my two images in a table with 1 row and 2 data cells so they'd appear next to each other, but you don't have to.

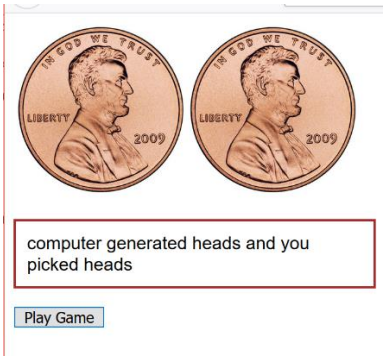
Step 11: And back to the function PlayGame() – add a prompt at the very top of the function that asks the user, "pick heads or tails"). Save the response in a variable (a new one, not the x or y you've already used).

Step 12: Now add a new if condtion either above or below the one you created in steps 6 and 7. The if condition should work as follows: if the new variable from 11 holds 'heads', change the src of the image you created in step 10 to the coin head, else change the src to the coin tail.

Step 13: Add to the document.getElementById for the paragraph (that currently writes out, 'computer picked '+y) what the user picked as well. My line looks like this (my paragraph's id was 'p1' and my user variable was z):

```
document.getElementById('p1').innerHTML = "comp generated " + y + " and you picked " + z
```

Step 14: Save and test. Now 2 images should show up – one representing the computer's random number of 0 or 1, with 0 being a coin head and 1 being a coin tail, and one representing what the user picked. In addition, the paragraph should print out what both picked. Mine looks like this (with a bit of style added to the paragraph to give it a border, and both the user picking heads and the computer generating the random number 0):



Step 15: let's indicate whether you won or not: In the function, add another if condition. This one should be below all the if conditions you've created so far. It is checking to see if what the user picked is what the computer generated. So if what the computer generated was heads, the y variable will hold heads, and if the user chose heads, the variable (my z variable) will hold heads, so they'll both be the same (y === z). If the computer generated tails, y will hold tails and if the user chose tails, the variable (my z again) will hold tails, so, again, y will be equal to z. So we can add a condition below all the ifs and around the document.getElementById() as follows:

```
if (y === z) {
        document.getElementById('p1').innerHTML = "You won!  Computer generated " + y + " and you picked " + z
}
else {
        document.getElementById('p1').innerHTML = "You lost.  Computer generated " + y + " and you picked " + z
}
```

What this does is check to see whether what the user picked and what the user generated were the same. If they're the same, it prints to the paragraph, "You won! …"

Otherwise it prints out, "You lost. …"

Almost done this part!

Global Variables:

Have you noticed that we keep using the same variable and parameter names in each function? We keep using x, y, z, and par again and again in every function we write. That's because variables within a function only belong to that function. The best way I have of describing this is that when I say, "professor" of CISC103, and "professor" of ENG110, the professor is unique to the class. The variables x, y, z, etc. are unique to each function, and thus the value inside of x in one function doesn't carry to another function. Equally, every time you click on a paragraph or button that calls that function, the variables are made fresh, as if they didn't exist before. To show you what I mean, try the following:

Step 16: Add another paragraph to your html code below the 'p1' paragraph. I gave this paragraph an id of 'wins' and also started out the paragraph's text with "Win Count: " (you'll change this in the .js playGame() function).

Step 17: in the function playGame(), at the very top of the function, add a variable wincount (yep, I named this variable multiple letters. I want to be clear what this variable holds, so I actually gave it a name that reflects what its purpose is inside the function. Initialize the wincount variable to be 0. The top of my function looks like this:

```
function PlayGame(par) {
        wincount = 0
        x = Math.floor(Math.random() * 2)
        z = prompt("Guess heads or tails")
        …
```

Step 18: In the if condition that checks whether you've won or lost (from step 15, above), if the user won (or if (y===z)) increase the count of the wincount variable as follows:

```
wincount = wincount + 1
```

Step 19: below the if condition (the very last line of the function, right above the function's closing }), change the paragraph with the id 'wins' that you created in step 16 using

```
document.getElementById('wins').innerHTML = "Win count: " + wincount
```

Step 20: Save and test your code by clicking again and again on the button in your html code. While there should be a paragraph that says “Win count: 0” or “Win count: 1” in the browser, this shouldn’t be working the way you want it to.

What you’d really like is for the win count to keep going up every time you win. It isn’t because every single time you click on the button on your web site, the function PlayGame() is called and the very first thing that happens is that wincount is created FOR THIS FUNCTION ONLY and set to be 0. Every. Single. Time. So if you win, the newly created wincount, which was set back to 0, is then increased to 1, and you get “Win count: 1” . If you didn’t win, wincount isn’t increased, which means you’ll get, “Win count: 0”, even if you’ve won 20 times before!

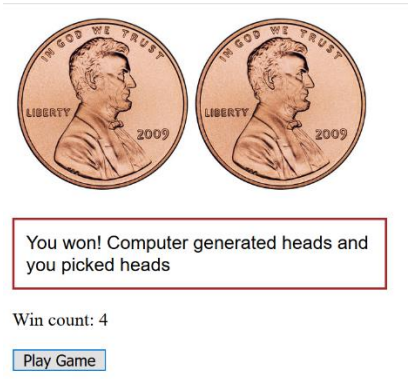
The problem is, wincount belongs to one calling of the function PlayGame. If that doesn’t make sense, realize that every time you call the function, wincount gets set to 0 again. That’s not what we want.

We want wincount to continue existing even when the PlayGame is done running each time. Equally, we want to originally set wincount to 0, but then every time we call PlayGame, we don’t want it to be reset to 0. In order to do this, we can make the variable OUTSIDE of any function, at the top of the .js file. So it would look like this:

```
wincount = 0
function PlayGame(par) {
        x = Math.floor(Math.random() * 2)
        z = prompt("Guess heads or tails")
        …
```

Step 21: Place the wincount variable outside and at the top of your javascript file, and set it to 0 (as demonstrated above).

Step 22: Save the code, and test it by clicking on the button on your web page multiple times. After playing 7 times and losing 3 times and winning 4 times, my results now look like this:



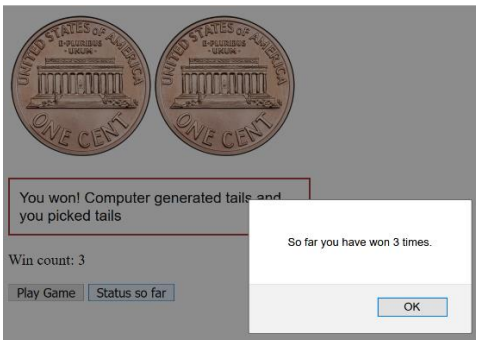
To show that you can use the variable wincount in other functions, let’s create another function as follows:

Step 23: in your html code, add a button below the Play Game button that has as a value, “Status so far” and using onClick, calls a function, Status(‘a’).

Step 24: In your .js file, create a second function below the PlayGame function called Status(par). In the function, use an alert box to print out the current number of wins as follows:

```
alert("So far you have won " + wincount + " times.")
```

Step 25: Save your code and test it by clicking on the Play Game multiple times. After you’ve played a couple of times, click on the Status so far button. An alert box should pop up showing you how many times you’ve won so far. Mine looks like this:



This shows that we can use global variables, or variables created above all the functions, in every function below it. Here we've used wincount in 2 separate functions, and it didn't lose its value between functions.

Congrats! You just wrote your first game

Your Turn:

1. Make sure you have the above html code and .js functions running and you understand it!

/**/

Random numbers with width and height:

Back to random numbers: there are so many fun things you can do with random numbers, let's just try a few more.

Step 1: Add a new image to your web page of something that moves (mine was a ghost). Set the width and the height the newer way, and make sure to give the image an id. Mine was an image of a ghost, and it looked like this:

```
<img src = "Images/ghost.jpg" height = 200 width = 200 alt = "a ghost pic" id = "g1">
```

Step 2: Add a button below the image. Make the button call the function, ChangeSize('g1') when it is clicked on, and make sure it sends in the id of the image for the parameter to hold.

Note: You may wish to position the button absolutely down from the top 400 px and in from the left 20 px, so that when the image changes size, the button doesn't jump all over the place. Give it a z-index of 10 so that the button is always in the front of anything else on the page.

Step 3: Save and make sure both the image and the button show up in the web site.

Step 4: In your .js file, create a function ChangeSize(par). In the function, generate 2 different random numbers between 10 and 350 (so you'll generate a random number up to 340, and then add 10. Make sure each random number is in its own variable.

Step 5: Using document.getElementById(par).width, set the width to be the first random number, and using .height, set the height to be the second random number.

Step 6: Save the .js file. Load the html file into the web browser and click on the button. Your image should randomly resize. Click again. It should randomly resize again.

/**/

Note: The following exercises will (if you so choose) be part of your final project. As such, you will probably want to create a new html page and a new .js page for these exercises

Moving position:

In tutorial 3, you systematically changed the size of an image by adding or subtracting 10 to its current size each time. For this exercise, you will be systematically moving an image by adding or subtracting 10 to its current location.

Step 1: in your html page, add an image of something that moves that can be the hero character in your final project (that will go around the page and "catch" things (mine was a zombie, but you can pick any themed hero you want). Since images belong inside of another tag, place it inside of a div tag and give the div an id ('z1' in my case) and give the div an in-line style.. The style should include positioning the image absolutely. Start by positioning it 0 px down from the top and 0 px in from the left (do an in-line style for this).

Step 2: add a button to the html page, outside of the div, and, again, position it absolutely (approximately 400 px from the top and 150 px in from the left, but you can always adjust this if it ends up under something) and a z-index of 11. (you can always play with all the numbers if you're not happy with any of them) Make this function call the function MoveRight('z1') when clicked on .

Step 3: Save your html and load it to see what it looks like so far. Make sure both images show up.

Step 4: In your .js file, create a function MoveRight(par).

Step 8: Still in the html, create yet another button (either position it below the 2 already on the page, or next to the two already on the page). This button calls the function MoveRight('z1') when clicked on, so make the button's value be 'Right'.

Step 9: In the .js file, first, above the functions at the top of the file (where you put the variable wincounts in the first part of this tutorial), create variable hleft and set it to the same number you positioned the hero div in from the left in step 1. (i.e., if I positioned my zombie 20 pixels in from the left absolutely in step 1, I want to set hleft=20) Now the variable hleft can be used by more than one function.

Step 10: Now create a function MoveRight(par).

Step 11: Add 10 to the hleft variable

```
hleft = hleft + 10
```

Step 12: now use document.getElementById(par).style.left =hleft to reset the image's position over from the left. Altogether my function looks like this:

```
function MoveRight(par) {
        hleft = hleft + 10
        document.getElementById(par).style.left = hleft + "px"
}
```

Step 13: Save this and then test it by clicking on the left button. Your hero image should move consistently to the right.

Step 14: Let's repeat the process for a MoveLeft function. First, create yet another button that calls a function MoveRLeft('z1') in your html file.

Step 15: In your .js file create a function MoveLeft(par). The function MoveLeft should be EXACTLY the same as the MoveRight function, with one difference. (NOTE: You DON'T need to create a new variable for hright – we're just going to subtract 10 from hleft to move to the left. So moving to the right means adding 10 to hleft, and moving left means subtracting 10 from the hleft pos. The line hleft = hleft + 10 should be changed to hleft = hleft - 10. This way you're moving over 10 pixels to the left each time.

Step 16: Save this and test all your functions.

Your Turn:

Repeat steps 1-16 for MoveUp and MoveDown (you'll need another global variable for htop)

****

Random numbers with position:

Step 1: Add another image of a goodie that you'd want to catch in your final project – it should thematically go with whatever hero you picked above, so my zombie wanted to catch chipmunks and eat them (so my image was a chipmunk) with the id (mine was 'c1' and give it an in-line style.. The style should include positioning the image absolutely. Start by positioning it 0 px down from the top and 0 px in from the left (do an in-line style for this).

Step 2: add another button, and, again, position it absolutely (maybe 400 px from the top and 150 px in from the left, with a z-index of 11). Make this function call the function ChangePos('c1') when clicked on (this button will eventually go away in the final project – we're using it to test the ChangePos function)

Step 3: Save your html and load it to see what it looks like so far. Make sure the images are showing up.

Step 4: In your .js file, add 2 global variables at the top of your js file for the goodie's top and left position (mine were ctop and cleft). Set them to 0 and 0, just like you positioned your goodie in step 1.

Step 5: Create a function ChangePos(par). In this function, generate a random value between 0 and 800 and place it in the global variable for the goodie's left position, e.g., cleft = Math.floor(Math.random()*800

Step 6: Set the left position to be the random a value as follows:

```
document.getElementById(par).style.left = cleft + "px"
```

Step 7: Save. Load into a browser and test by clicking on the second button. Your goodie should move left and right all over the page.

Step8: within the ChangePos function, repeat steps 5-7 by generating another random number between 0 and 800 and placing it in the goodies' top variable, then use document.getElementById to set the top position to the new random number.

Step 9: Save and make sure that each time you click on the button, the goodie moves all over the screen.

It's getting a bit fun, huh! What if you only wanted the image to move in one direction?

Combining moving and changing position randomly:

Step 7: In your html code, Add yet another image of something ucky – this will be the thing that destroys your hero – (a bad thing - a car, a rock, a shark – mine was a bunny monster that will destroy my zombie if it runs into the zombie). Using an in-line style, position it absolutely 0px down from the top and 0px in from the left.

Step 8: Still in the html, create yet another button (Again, this one will go away, so position it wherever it's convenient for now). This button calls the function MoveAcross('m1') when clicked on, (the bunny monster image has an id of 'm1', so that's where the m1 came from)

Step 9: In the .js file, first, at the top, create a global variable mleft for the ucky thing's position over from the left. Now the variable k can be used by more than one function.

Step 10: Now create a function MoveAcross(par). Inside the function generate a random number between 0 and 25 and put it into a variable x.

Step 11: Add that random number x to mleft as follows:

```
mleft = mleft + x
```

Step 12: now use document.getElementById(par).style.left =mleft to reset the image's position over from the left. Inside the function, you'll have something like this:

```
x = Math.floor(Math.random() * 25) + 1
mleft = mleft + x
document.getElementById(par).style.left = mleft + "px"
```

Step 13: Save this and then test it by clicking on the left button. Your image should move consistently to the left.

Step 14: Add a condition that if mleft gets to be greater than some number (maybe 500), it should be set back to 0.

Step 15: Save this and test your function.

Your Turn:

1. Make sure you have the MoveLeft, MoveRight, MoveUp, MoveDown, ChangePos, and MoveAcross functions working!

****

setTimeout:

So far the user has had to click on something to have something happen. But javaScript allows some things to happen automatically, and allows them to happen again and again. This is done using setTimeout.

What setTimeout is: It's a way of making a function happen without actually having to click on it.

What it looks like: **setTimeout(function(){**nameofafunction(par)**;},**3000**);**

The parts you have control over are the **nameofafunction(par)** and the **3000.**

Nameofafunction(par) is the name of a function that you want to be called automatically instead of having to click on a button to make the function happen (note that you'll have to click the first time).

The **3000** is the amount of time **javaScript** pauses before calling that function. Remember, computers are insanely fast.

To see how this works, try the following (this is not part of the final project, but you should add it for now to see how setTimeout works)

Step 17: To your javaScript add the following 2 functions:

```
function setToRed (par)
{       document.getElementById(par).style.color = "#FF0000"
}
function setToBlack (par)
{       document.getElementById(par).style.color = "#000000"
```

```
}
```

Step 18: Now in your html,a paragraph. Give the paragraph the id of 'p8' (because I know we haven't used that before). Make sure the paragraph has some actual text.

Step 19: Add 2 buttons. Make the first button work such that when you click on it, it should call setToRed('p8'). Make the second one work such that when you click on it, it should call setToBlack('p8').

Step 20: Save all. Load the html into your browser. Click on the red button first. The paragraph's text should turn red. Now click on the black button – the paragraph's text should turn to black.

You can keep clicking on these buttons again and again to keep turning the text red and then black. But you can do it automatically using setTimeOut as follows:

Step 21: in the setToRed(par) function, make the last line (the line before the }) be setTimeout as follows:

```
function setToRed(par)
{       document.getElementById(par).style.color = "#FF0000"
        setTimeout(function(){setToBlack(par);}, 2000)
}
```

Step 22: Save and test. What should happen now is you should click on the red button. The text will turn red. Then after 2 seconds, ALL ON ITS OWN, the text will turn back to black. That's because setTimeout calls the function setToBlack after 2000 msec.

Step23: Change 2000 to 500 and try this again.

Step 24: Change the 2000 (or, after step 23, the 500) to 8000 and try again.

Each time you change that number, you're changing the amount of time javaScript pauses before calling the function setToBlack. You can make the change slow or fast by changing that number.

You can also make it blink, simply by adding a setTimeout function call to the setToBlack function as follows:

Step 25: Add the setTimeout to the setToBlack function as below:

```
function setToBlack(par)
{       document.getElementById(par).style.color = "#000000";
        setTimeout(function(){setToRed(par);}, 2000)
}
```

(Make sure you change the setToRed function's setTimeout time delay back to 2000 as well)

Step 26: Save and test by clicking on the "turn to red" button once and then watch it blink (albeit slowly!)

Moving Automatically

Let's go back to the MoveAcross function (steps 7-15 on page 7). Let's modify this to have the image move on its own automatically using setTimeout.

Step 27: Add a setTimeout to this function so that it keeps moving automatically. The line should be the last line in the function (right above the closing squiggly bracket and after the if condition closes) and should look as follows:

```
setTimeout(function(){MoveAcross(par);},500)
```

Step 28: Save and test. You should be able to click on the button that called MoveAcross and watch the image move automatically across the page. If you want it to move faster, change the pause time from 500 to 300 (or shorter) and try again.

Cool, huh!

Your Turn:

1. Get MoveAcross working.
2. Modify the ChangePos function by adding a setTimeout function at the bottom that calls ChangePos every 10 seconds
3. Add to the MoveAcross function the following:
    a. an if condition (above the setTimeout function but below everything else). The if condition will work as follows: if the bad thing is at the edge of the screen, (maybe 800), it's left position (the mleft) should be set back to 0 so it starts over at the left side.
    b. Once you have that working, you're going to have the monster move to different places vertically.
        i. Add a mtop global variable at the top of your js file. Set it to any number you want.
        ii. Now ***inside the if condition*** (a) generate a random number between 0 and 800 (you can play with this number later) and set the mtop variable to hold this random number:

            document.getElementById('m1').style.top = mtop+"px"

iii. Save this and test it. What should happen now is when the monster gets to the edge of the page, it should go back to the beginning of the page and start over, only at a different position down vertically from the top.

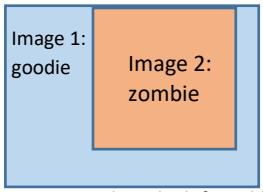
/**/

Finally, if two things are in the same place:

Now that you can position things absolutely, and randomly, using javaScript, it may be useful to determine if two elements (e.g., images) are on top of each other.

To see if images are on top of each other, you need to see if the position in pixels from the left of one image is the same as the position in pixels from the left of the second image. (You'll also need to see that their position from the top is the same for both) In the code so far, the variable hleft holds how far the one image (the zombie image) is over from the left, and the variable cleft holds the goodie's position over from the left in pixels.

Now, if the goodie and the zombie images are in exactly the same place in terms of how far they are from the left, hleft and cleft will both be the same (hleft===cleft). But here's the thing. Hleft and cleft are in pixels, and pixels are seriously small. We may want to consider 2 images over each other if they're relatively close to each other. For instance, if you have the following:



In this case, even though cleft and hleft aren't exactly the same, I'd consider the zombie and the goodie to be in the same place. To accomplish this, you will want to check if cleft is close to hleft, and, if so, then it would be considered on top of each other.

Step 33: In your MoveRight function, add a check to see if a and k are close to each other as follows: This should be right above the functions closing }

```
if ((hleft> cleft-10) && (hleft < cleft + 10)) {
        alert("bunny killed zombie!!")
}
```

Now when hleft is within 20 of cleft (say hleft is 40, then cleft can be anywhere 30 (40 – 10) to 50 (40 + 10).

Step 34: IN your MoveLeft function, add the same check.

Step 36. Save. Click on the button that calls ChangePos. Then move the zombie near the goodie by using the MoveLeft and the MoveRight buttons. When the zombie gets close to the goodie, you should get an alert box popping up.

Your Turn:

1. Right now the MoveRight function only checks to see if the two images are close to each other's left position. Add an if condition (within the left if condition, above) that also checks to see if the two image's top positions are within maybe 20 pixels of each other.
2. Add the same checks to the MvoeLeft, the MoveUp and the MoveDown.
3. Scoring: First, In your html code add a paragraph, and give it an id.
   Now, you had a wincount variable for the coin toss game. Add a similar global eatcount variable to your game. Then, where the alert is in the MoveLeft, MoveRight, etc. functions, add eatcount = eatcount+1. Change the paragraph in your html code's innerHTML to the new eatcount so that every time the zombie is over the goodie, the current number of goodies the zombie has eaten is displayed.

Your game is getting there…