JavaScript: Lab 3: Video Tutorials3

(100 pts, due Mon, May 3)

From getElementById to Get Info:

1. Getting Alt (5 pts)
2. Getting BorderColor (8 pts)
3. Changing Style based on Previous Style (8 pts)
4. Switching Back and Forth (9 pts)

From innerHTML 1:

1. Get all of the above working (12 pts)
2. Changing Image, Paragraph and Style (12 pts)

From Joining Strings Together:

1. JoinString (4 pts)
2. JoinString 2 (4 pts)
3. AddNums (4 pts)
4. GetPersonalInfo (4 pts)
5. Multiplication (8 pts)

From innerHTML 2:

6. Change Entire List (3 pts)
7. AddNewItemToList (4 pts)
8. ChangeDiv (6 pts)
9. Get Above to work (4 pts)
10. Choosing Old List (4 pts)

Note: these are tutorials that REALLY CLOSELY parallel week 10's videos/ppts. Please watch week 10's videos for this lab.

Contents

Part 1(30 pts total):getElementById for getting information: 2
Getting Alt (5 pts) 2
Getting BorderColor (8 pts) 3
Changing Style based on Previous Style (8 pts): 4
Switching Back and forth (9 pts): 4
Part 2: getElementById innerHTML (1) (24 pts) 4
Your Turn: 7
1) (12 pts)Get all of the above working 7
2) (12 pts) Changing image, paragraph, and style: 7
Part 3(24 pts) : Joining Strings Together 7
Your Turn (4 pts) JoinString 8
Your Turn (4 pts) JoinString2 9

Your Turn: 10
1. Get JoinStrings1 working (4 pts) 10
2. Get JoinStrings2 working (4 pts) 10
3. AddNums (4 pts) 10
4. GetPersonalInfo(4 pts) 10

5. Multiplication (8 pts)..... 10
Part 4: innerHTML 2 (21 pts total)..... 10
Your Turn: (3 pts) Change entire list: 12
Your Turn:..... 13
1. AddNewItemsToList(4 pts):..... 13
2. ChangeDiv (6 pts):..... 13
Your Turn..... 16
1. (4) Make sure the code above works..... 16
2. (4) Choosing Old List: 16

Create a new HTML and .js file for this lab. It will reduce cluttered web pages when things don't work so it will be easier to find

Part 1(30 pts total):getElementById for getting information:

You have used document.getElementById to change existing information

However, you can also use document.getElementById to obtain information about your html page. Let's try it:

We're going to use Javascript to read an alt tag from a particular image off your web page

Getting Alt (5 pts)

Step 1: create a new function in your .js. Call it getAlt(par).

Step 2: Inside the function, use a variable to hold what the alt of an image is by using document.getElementById(par) as follows (the variable will be called x):

```
x = document.getElementById(par).alt
```

Explained:

Here: document.getElementById is on the RIGHT, which means that whatever value it holds will be put into the variable on the LEFT.

This is similar to the prompt box:

```
x = prompt("Enter a color")
```

Where whatever the user types into the prompt box ends up in x. It doesn't work the other way around like this;

```
prompt("Enter a color") = x  /* THIS DOESN"T WORK!!*/
```

Because in this case you're taking what is in x and trying to put it into the prompt box, and with prompt boxes that doesn't work.

The rule is: the value on the right goes into the value on the left, e.g.,

```
x = 3
x = prompt("enter a number")
x = document.getElementById(par).alt
```

In each of the above examples, the value on the right goes into the variable x on the left.

Step 3: to show what x holds, use an alert box as follows:

```
alert(x)
```

Putting this all together, you'd have:

```
function getAlt(par) {
        x = document.getElementById(par).alt
```

```
        alert(x)
}
```

Step 4: In your html page. Add 3 images to the page, making sure each has a unique id and an alt tag. Add an onclick to each image that calls the above function (with the id of the image being passed in to the function's parameter). Make sure the id is in single quotes.

Step 5: Save both files. Load the html page into a browser and test it by clicking on each of the buttons. When you click on the buttons, you should have an alert box that pops up with the image's alt tag.

Getting Style:

You can use document.getElementById to get style from tags as well. As long as the html tag has that style set (using CSS), you can get it and put it into a variable using getElementById. So, for instance, you can do the following:

```
x = document.getElementById(par).style.borderColor
```

and now x holds the html element's border color.

Let's try it:

Getting BorderColor (8 pts)

Step 1: create a new javascript function – it will be very close to the getAlt function you just wrote. The differences?

a) The name should be something like, getBorderColor.
b) Instead of using getElementById to get the alt, use it to get the borderColor as shown above.

Step 2: In your html page, either add 3 more images or copy the above images and paste so they're in your web page twice.

Step 3: For your second set of images, give each a unique id (so if you copied the image from above, change this set of images' ids)

Step 4: For this second set of images, give them an inline style that includes a border color. To give you an example, one of my images looks like this:

```
<img src = "Images/bee.jpg"
        id = "bee2"
        alt= "a bee"
        style = "width:100px; height:100px;
        border: 6px solid green;
        border-radius: 25px; box-shadow: 8px 8px 5px gray;">
```

Step 5: Give each of the 3 images a different border color.

Step 6: Add a call to the function getBorderColor, with the id being passed to the function's parameter, using onClick

Step 7: Save it and test it. When you click on one of the second 3 images, its border color should pop up in an alert box.

This doesn't just work for images – it will work for any element on a web page that has a border color.

Step 8: In your html page, add 3 paragraphs.

Step 9: Give each paragraph its own unique id

Step 10: Give each paragraph an inline style with a border color. While the colors don't have to be the same as the 3 images above, for this exercise give them the same colors. One of mine looked like this:

```
<p id = "p1"
        style = "width:100px; height:100px; padding: 8px;
        font-family: Arial; font-size: 22px;
        border: 6px solid green;
```

border-radius: 25px; box-shadow: 8px 8px 5px gray;"> Bees Para </p>

Step 11: Add an onClick call to the same function you called to get the border color above, making sure you pass the paragraph's id into the function's parameter.

Step 12: Save the html file and load it into a browser. When you click on each of the paragraphs, the alert box should pop up with the color of the paragraph's border, just like it does with the images.

Changing Style based on Previous Style (8 pts):

We can use this information to change the border color, based on the current color, using an if condition.

Step 13: In the getBorderColor function, add an if condition as follows: If the border's color is the color of the first image, change it to the color of the second image. If it's the color of the second image, change it to the image of the third image. If it's the image of the third image, change it to the border color of the first image (See video).

Step 14: Save and test. When you click on the images, their borders should change. Click on them again. Click on them a third time. They should be back to where they started.

Step 15: Try clicking on the paragraphs. The same thing should happen as what happened to the images.

Switching Back and forth (9 pts):

You can use the current state to switch the styles back and forth. If the style is in one state, you can switch it to another state. Otherwise if it's in the other state, switch it back to the first state. For this, I set a bunch of style elements, including the border color for a paragraph. For mine, I created a paragraph with a lightgray border, a relatively small font size, a light background, a gray font color, a small amount of padding, and a relatively small width and height. The paragraph calls a function on click.

The function checks to see if the border color is light gray, and if it is, it set the border color to be a bright color. It set the background color to be a high contrast color, and the font's color to be another high contrast color. It set the font's size to be a large one, and to be bold, and it set the width and height to be relatively large.

So if you click once, the paragraph will change to have the bold style described above. But I want to be able to click again to switch the paragraph's style back to the plain, starting style.

So within the same function, I have another if condition that checks to see if the border color is the bright color I set it to above. If it is, I pretty much repeated the first if condition, only with very mild, unassuming sizes, colors, and widths and heights. So, for instance, I set the background color to be white, the font's color to be gray, the font's size to be 14px, etc.

Step 16: Get the above working so that if you clickon the paragraph once, its style changes one way, and if you click it again, it switches back.

Your Turn:

Get working the following:

1. GetAlt (5 pts)
2. GetBorderColor (8 pts),
3. Changing style based on previous style(8 pts), and
4. switching back and forth (9 pts)

Part 2: getElementById innerHTML (1) (24 pts)

So far we've changed the style of something with an id, the src, the width, the height, and the alt of an image, but we haven't changed the actual content of your html page.

Changing the content can be done using innerHTML

First, you must understand exactly what innerHTML is.

The **innerHTML** is what is between the opening and the closing tag of anything, regardless of what the tag is.

So in the following html code:

 This is the innerHTML text between the opening and closing tag

The innerHTML of firstp is: “This is the innerHTML text between the opening and closing tag”

In this html code:

<h1 id = “largeheader”>**Autumn**</h1>

The innerHTML of largeheader is “Autumn”

In this html code:

link to cows are awesome

The innerHTML is “link to cows are awesome”

Start Reading innerHTML:

Step 1: in your html page, add 3 paragraphs. Give each an id. Make sure the ids are unique. Using onClick, make each call the function, changeinnerHTML (that you will be writing shortly) with the paragraph’s id as a parameter. For now, make the content of the paragraph short – a one-word thing, like a season, an animal, a holiday, something like that. So, for instance, one of the 3 paragraphs might looks like this:

```
<p id = 'p1' onClick = "changeinnerHTML('p1')">Cows</p>
```

Step 2: Save your html page. Load it in a browser just to be sure it’s working.

Step 3: in your .js file, add a function readinnerHTML(par). The function will read the innerHTML of whatever id you sent into the function. And then you’ll use an alert to print it out (for now). So the function will look like this:

```
function changeinnerHTML(par) {
        document.getElementById(par).innerHTML = "Horses"
}
```

Step 4: Save the .js file and test the function by clicking on each of the paragraphs.

What happened?

1. When you clicked on the paragraph, you sent the id of the paragraph into the parameter par in the function in the .js file.
2. Then document.getElementById(par).innerHTML, which is currently “Cow”, gets set to what is on the right, or “Horses”
3. In the html page, the old innerHTML gets changed to “Horses.

Just like with the style and the image information, you can also read the innerHTML from the web page. You just need to put the innerHTML on the right and a variable on the left. So you could modify the above function as follows:

```
x = document.getElementById(par).innerHTML
```

and then have an alert button popping up with the innerHTML content as follows:

Step 5: In your function, above the line used to change the innerHTML to be Horse, add the line that reads the innerHTML (just like the line above).

Step 6: right below that, add an alert that shows the content of x

Step 7: In your html page, add 2 more paragraphs. Mine were “Giraffes” and “Pandas”.

Step 8: Give each paragraph a unique id, and have it call the same function the Cows paragraph calls when they’re clicked on.

Step 9: Save and test. If you click on each of the paragraphs, its innerHTML content should pop up (Cows, Giraffes, Pandas).

You can make this more interesting by adding an 'if' condition to your function. The if condition will add more information about whatever the user clicked on.

Step 10: To give you an example, I had as my paragraphs, a cow, a giraffe, and a panda. So my if condition looks like this:

```
if (x === 'Cows') {
        y = "You are more likely to be killed by a cow than a shark"
}
else if (x === 'Giraffes') {
        y = "A giraffe's legs are longer than most humans are tall, and their necks are too short to reach the ground"
}
else if (x === 'Pandas') {
        y = "A baby panda eats its mother's poop, and all pandas have 6 digits on their paws"
}
```

So the variable y will hold extra information about the paragraph the user clicked on.

Step 11: Finally, you will want to put the innerHTML together with the new information. I made a new variable to hold both strings combined, and included a ": " between the two so they're nicely separated:

```
z=x + ": "+y
```

So the modified function will look like this:

```
function readinnerHTML(par) {
        x = document.getElementById(par).innerHTML
        if (x === 'Cows') {
                y = "You are more likely to be killed by a cow than a shark"
        }
        else if (x === 'Giraffes') {
                y = "A giraffe's legs are longer than most humans are tall, and their necks are too short to reach
the ground"
        }
        else if (x === 'Pandas') {
                y = "A baby panda eats its mother's poop, and all pandas have 6 digits on their paws"
        }
        z=x + ": "+y

        alert(z)
}.
```

Step 12: save both the html and the .js, and load the .html into a browser and test it.

Changing the innerHTML:

Instead of using an alert box, you can also replace the existing paragraph with the new, expanded information. To do that, you'd replace the js code, alert(x + ": " + y) with

```
document.getElementById(par).innerHTML = x + ": "+y
```

Step 13: Modify your function above by replacing the alert box with the above document.getElementById.

Step 14: Save both the html and the .js code, and test this in the browser.

Make sure you understand what is going on!! Make sure you understand what x holds, and also that par holds the id of an element on your html page. The tag whose innerHTML is being read and the tag whose innerHTML is being changed are both identified by the id in the par parameter.

Concepts to remember:

1) What is on the right side goes into what is on the left side.
2) If you want it literally, you put quotes around it. If you want what it holds (if it's a variable or a parameter, don't.
    a. The exception to that rule is numbers
    b. E.g., if x holds 'p1' and 'p1' is an id of a paragraph, then you'd either want:

document.getElementById('p1').innerHTML or
document.getElementById(x).innerHTML

In step 13, we want what's inside of x and what's inside of y to show up, but we literally want the : to show up, so the : goes in quotes, and the x and y don't, hence:
document.getElementById(par).innerHTML = x + ": "+y

3) In the javascript functions, you know which element is being read or changed based on the id between the two parentheses. getElementById has to have the id of some element on your web page between the two parentheses.
4) Id in getElementById is a capital I and small d, like Igore dances, not llama determines.

You can use document.getElementById to modify another paragraph on your web page.

Step 15: Add both an image and another paragraph above the 3 paragraphs you created in step 1, above. Give the paragraph an id of 'mainid'.

Step 16: modify the bottom document.getElementById() that changes the innerHTML (not the one that reads the innerHTML). You are going to change it so that, instead of modifying the paragraph that calls the function, you modify the mainid paragraph simply by changing the innerHTML of that bottom document.getElementById(). So that bottom line will now look like this:

```
document.getElementById('mainid').innerHTML = x + ": "+y
```

Step 17 Save your html code and your .js file. Load the html code and click on each of the paragraphs (not the mainid paragraph). When you click on a paragraph, the mainid paragraph should change text to reflect the paragraph you clicked on.

Your Turn:

1) (12 pts)Get all of the above working
2) (12 pts) Changing image, paragraph, and style:

In step15, you added an image to the html code. Give the image an id. You can now use document.getElementById() with the id of the image to change the .src of the image, along with the mainid's paragraph.

Modify the function you created above so that, depending on what paragraph is clicked on, an appropriate image shows up as well as a longer paragraph. Modify the style of the paragraph so that it fits with either the image's colors or the paragraph's themes (so, for instance, if you've got the word Panda as a paragraph, when you click on it, you want a picture of the panda to pop up in the image's spot, text about pandas to pop up in the mainid paragraph, and maybe give the mainid paragraph a background of bamboo with a yellow-green border and yellow text, or something like that).

Note that to do this, you'll have to modify each of the if conditions so that each one uses document.getElementById to change the image's src and also to change mainid's style.

Part 3(24 pts) : Joining Strings Together

Javascript considers anything between quotes to be a string. The quotes can either be double quotes or single quotes. So a string is

- "cat",
- "p1"
- "This is a string"
- "10"

Joining Strings:

Yep, if you put quotes around it, it's considered a string, even if what is between the quotes is a number.

In previous exercises, you've joined 2 different strings together into a new string using the + sign as follows:

```
x = "<p>"
y = "dog"
z = "</p>"
q = x + y + z
```

Adding to the end of a string

But you can also just add on to the end of a string. So, instead of a variable being something new, you'd say:

```
x = ""
x = x + "<p>"
x = x + "dog"
x = x + "</p>"
```

x = ""

What you've just done is initialize x originally (the first line) to hold no characters. In computer science, there are many times when you must initialize a variable, but you may not want to give it a value right off the bat. So instead you'd initialize it to be blank, empty or 0 (if it's a number). In this case we're starting x out as holding an empty string of characters.

x = x + " "

Next you've joined on to the end of x the ** **. When you do that, you join to the empty string. So now x holds ** **.

x = x + "dog"

Next you join dog onto the end of x. So you're joining **dog** onto , resulting in ** dog**

x = x + " "

And next you're joining ** ** onto the end of x, so you'd end up with ** dog ** inside of x

3) Let's try it:

Step 1. Create a new function in your javaScript. It should look like this:

```
function joinString(par) {
        x = ""
        alert("x currently holds: "+x)
        x = x + "<p>"
        alert("x currently holds: "+x)
        x = x + "dog"
        alert("x currently holds: "+x)
        x = x + "</p>"
        alert("x currently holds: "+x)
}
```

Step 2: Add a paragraph to your html code. Give the paragraph an id, and make the paragraph call joinString function with the paragraph's id. Note: you're not really doing anything with the paragraph in this function.

Step 3: Save all and test this. Do you see how each time you add to the end of x, x changes what it holds?

Your Turn (4 pts) JoinString

- Get the above working.

You could use a prompt to get a variable, and then join that to a variable:

Step 4: in the above function, change the alert lines to:

```
y=prompt("Enter a pronoun")
z = prompt("Enter an adverb")
q=prompt("Enter a verb")
r = prompt("Enter a noun phrase")
```

Step 5: join each of the variables with x as follows:

```
function joinString2(par) {
        x = ""
        y = prompt("Enter a pronoun")
        x = x + y
        z = prompt("Enter an adverb")
        x = x + z
        q = prompt("Enter a verb")
```

```
        x = x + q
        r = prompt("Enter a noun or noun phrase")
        x = x + r
        document.getElementById(par).innerHTML = x
}
```

Step 6: Save and test. Right now, all the words might be strung together unless you added spaces when you typed in words (which most people won't do). To fix that:

Step 7: join a space after each word as follows:

```
x = x + y + " "
```

for each time you join a new variable to x, you want to join a space as well.

Step 8: Save and test. There should now be spaces between the prompted words

Your Turn (4 pts) JoinString2

- Get the above working.

Numbers versus Strings of characters

Notice how you are using the + sign and it is joining strings together. But if you had numbers, you'd want the numbers to be added together. This is called operator overloading, and it means that the same operator, the + sign, does something different with strings of characters (or words) than it does with numbers.

So if you have x = "hi" + " there"

X should hold "hi there"

But if you have x = 3 + 4

X should hold 7, and not "34"

So + works differently with numbers and with strings. And if there are ""around something, it's a string, even if the quotes are around a number.

So x = "3" + "4"

Now x will hold "34"

Try it.

Step 9: Make a copy of the function joinString2.

Step 10: Change the name to addNum

Step 11: set x to be 0 to start with:

```
x=0
```

Step 12: Change the prompt to be, ("Enter a number") for all of the prompts

parseInt

You need to tell the computer that the numbers you are reading in are numbers, and not strings. So you will need to add parseInt (that's an I as in Iguana, not an L as in Llama) to each prompt:

y = parseInt(prompt("Enter a number"))

This converts a string of characters to an integer. Think of texting – if you texted "I'm 2 tired 4 work", those numbers are really being used as a word, or string of characters. But if you wanted to add, you'd use 2 differently. The computer sees 2 and "2" as two different entities. In order to be able to tell the difference between those 2, parseInt clarifies that the number should be used as number.

Step 13: place parseInt around each of the prompts.

Step 14: In your html, create a new paragraph with an id, and have that paragraph call this function with its id going into the function's parameter

Step 15: save and test

Your Turn:

1. Get JoinStrings1 working (4 pts)
2. Get JoinStrings2 working (4 pts)
3. AddNums (4 pts) Make sure you've got addNums working.
4. GetPersonalInfo(4 pts) Write a function that uses a prompt to ask the user their first name,one that asks their last name, one that asks their year (sophomore, senior, etc.), and one that asks their major Join the answers together into a nicely formatted string (with spaces and commas) and print it out using document.getElementById. Make sure you include a paragraph in your html code that has an id, and calls the function with its id.
5. Multiplication (8 pts) In your html create a table. The table should have at least 4 data cells (each with their own id) and each with a number in it.
   Each data cell should call a function, mult() with its id.
   Below the table create a paragraph with an id. This paragraph is where the calculated answer will go (as described below).

   In your js file, create a function mult that takes an input parameter. Use document.getElementById's innerHTML to read in the number that is in the data cell associated with the id the parameter holds (from the html code).

   Below that use a prompt to ask the user, "what would you like to multiply " + x + " by?" (assuming x is your variable that holds the innerHTML of the table cell you clicked on). The user should type something in, and that will go into a variable.

   Then multiply what the user types in with the number read in using document.getElementById (aka x in the above example).

   Finally, Print out the answer in the paragraph below the table using, again, document.getElementById and that paragraph's id.

Part 4: innerHTML 2 (21 pts total)

So far you've learned about using document.getElementById for both reading and modifying any element on your html page with an id. So you can read and modify style, image src, width, height, and alt info, and the innerHTML of elements with tags.

Let's go over exactly what the innerHTML is. The innerHTML of an element is all of the material between an opening and closing tag. So, in the following,

 This is the innerHTML text between the opening and closing tag

The innerHTML of firstp is,

This is the innerHTML text between the opening and closing tag

To be clear, though, the innerHTML is EVERYTHING between the opening and closing tag of the element with the id. So, for

 ** link to udel **

The innerHTML of linked would be

** link to udel **

Because all of that occurs between the opening and closing tag of the p with the id of 'linked'

Thus, to change the link to something else, you'd do the following:

document.getElementById('linked').innerHTML = "link to google"

You'd have to replace EVERYTHING between the opening and closing tag for the code generated by javaScript to work.

In the following,

```
<ol id = “list1”>
        <li id = “firstItem”> cats </li>
        <li id=“seconditem”> dogs </li>
</ol>
```

The innerHTML of **list1** is all of the following:

**<li id = “firstItem”> cats **
**<li id=“seconditem”> dogs **

And given the following html code,

```
<div id = "idofadiv">
        <h1> Horror Movies</h1>
        <p> I love crazy, schlocky, funny horror movies.  I don’t like gory slasher movies at all.  They’re just gross. I really like the horror movies that either are psychologically scary or are just silly.
        </p>
        <h2>B Horror Movies</h2>
        <p> There are funny horror movies, like Sean of the Dead and Zombieland, and then there are horror movies that are so bad they’re funny.  I like those too!
        </p>
</div>
```

the innerHTML of idofadiv would be all of:

<h1> Horror Movies</h1>
** I love crazy, schlocky, funny horror movies. I don’t like gory slasher movies at all. They’re just gross. I really like the horror movies that either are psychologically scary or are just silly.**
** **
<h2>B Horror Movies</h2>
** There are funny horror movies, like Sean of the Dead and Zombieland, and then there are horror movies that are so bad they’re funny. I like those too!**
** **

Using innerHTML to change an entire list:

Step 1: In your html page, create a list. In this case, make it be an unordered list. Make there be one or two list items in the list. And, of course, make sure you give the list (not the list items, but the list) an id.

Step 2: make a separate button that calls a function, changeList() . Pass the id of the list into the function’s parameter. Make the value of the button be, “change the list?”

Step 3: in a .js file, create a function changeList(par).The function will change the innerHTML of the list as follows:

```
x=”<li> ghosts</li>”
y = “<li> zombies</li>”
z = “<li>monsters</li>”
q = “<li>ghouls</li>”
```

and then

```
document.getElementById(par).innerHTML = x+y+z+q
```

So your function will look like this:

```
function changeList(par) {
        x="<li> ghosts</li>"
        y = "<li> zombies</li>"
        z = "<li>monsters</li>"
        q = "<li>ghouls</li>"
        v = x + y + z + q
        document.getElementById(par).innerHTML = v
```

```
}
```

Step 4: Save and test. The list should change to the updated list.

Troubleshooting:

- Make sure that the button in your html calls a function with exactly the same name as the name of the function in your javaScript. (so in the example above, small c in change, large L in list
- Make sure in the html you're passing in the id of the entire list (the ul) instead of an individual list item (an li)
- Make sure you have an opening and closing quote for each string on each line in the function that puts a string on the right into a variable on the left
- Make sure you've got opening and closing { and } around the entire function.
- Double check spelling of document.getElementById(..).innerHTML

Your Turn: (3 pts) Change entire list:

1) Get the above working

Note that I could have written out,

```
document.getElementById(par).innerHTML = "<li> ghosts</li><li> zombies</li><li>monsters</li><li>ghouls</li>"
```

This line and the code above do exactly the same thing – it's just easier to find problems in the code above than it is to find errors in this line.

Using getElementById to Add to an Existing List

So far, so good. But what if you want to add to the existing list?

Step 5: Create a new button in your html code. The value of the button should be: "Add to the List?" Have the new button call a new function (maybe addToList) and make sure you send in the id of the overall list into the functions parameter.

Step 6: In your function, create a new function with the same name as *step 5*'s function call (again, maybe addToList). Make sure the function has a parameter

Step 7: In the function add code that gets the current list into a variable as follows:

```
k = document.getElementById(par).innerHTML
```

This line gets the current innerHTML of your list. Thus k will now hold all of the innerHTML of your list, or all of the list items in there so far.

Variable Name Side Note:

Note that computer scientists often use single letters as variables. This is because WE ARE LAZY! It is much easier to use as a variable name, k, than it is to use, 'aVariableThatHoldsTheInnerHTMLOfTheList' Both are perfectly legitimate variable names, as is 'puppies'.

In general when programming, it is good coding practice to give variables short but at least somewhat descriptive names. So, for instance, idpar is a better name for a parameter that will hold an id than, say, x. idpar is at least a bit more descriptive, yet it isn't too likely that you'll misspell it.

That said, if you are ever reading code and you see a single letter (most commonly, x, y, z, i, j, and k) then it is most likely a lazy programmer naming their variables and parameters. But you should probably think about using better names when you're writing code.

Step 8: Now pretty much copy the guts of your changelist function. Paste it below the line in which you got the innerHTML of the list (in step 7)

Step 9: Modify the last line to be:

```
document.getElementById(par).innerHTML = k+ v
```

Step 10: Save your code and test it. Does the existing list now appear at the beginning of your new list?

Step 11: Change the last line to be as follows:

```
document.getElementById(par).innerHTML =  v + k
```

Step 12: Save the code and test. Do you see the difference between step 8 and this? Do you understand why this happens?

Add new content to list:

So far the code is relatively boring. Every time you click on the list, the exact same list is generated. Plus you're adding the same content again and again.

Let's make it more interactive by asking the user what they want to input into the list. To do this, you'd use a prompt to ask the user what item they want to add to the list, and a variable to hold what that item as follows:

```
m = prompt("What item do you want to add to the list?")
```

and then x would be:

```
x="<li>" + m + "</li>"
```

Make sure you understand this. In this line, m is the item the user typed in (say it's ghouls). But to make it a list item, we have to surround it with <li> and . Those are literal items – we literally want <li> and to be written to the html page. But we don't want m to literally be written to the page, we want what's inside of m. So m does not go in quotes, but <li> and do. And, of course, we have to join everything together. That's what the + do – join all 3 things together. And, finally, we want to put it all into the variable x.

You could also do it like this:

```
v= "<li>"
w = "</li>"
m = prompt("What item do you want to add to the list?")
x = v+m+w
```

That will result in exactly the same results.

Step 13: Add this to your function. Your function should look like this:

```
function changeList(par) {
        k = document.getElementById(par).innerHTML
        m = prompt("What item do you want to add to the list?")
        x="<li>" + m + "</li>"
        y = "<li> zombies</li>"
        z = "<li>monsters</li>"
        q = "<li>ghouls</li>"
        v = x + y + z + q
        document.getElementById(par).innerHTML = k+ v
}
```

Step 12: Save and test.

Your Turn:

1. ### AddNewItemsToList(4 pts):

   Right now, instead of the <li> ghosts that was in the original list, a prompt is used to get an item for the list, and that is added to the existing list. The other 3 items (zombies, monsters, and ghouls) are not items the user entered. We want the user to enter in all 4 items. So repeat the process for all 3 other list items so that the user is prompted for all 4 different list items.

2. ### ChangeDiv (6 pts):

   At the beginning of this tutorial, we discussed how the innerHTML is everything between two tags, including div tags. So if you set the innerHTML of a div, you can add a new src, paragraph, list, anything at all and that will show up between the div tags. Let's do that. On your web page, create a div with an id.

   a. Style the div by setting a width, a border, a background color, and setting the margin to be auto. (This isn't the innerHTML.

b. Add an image. You can do this by creating a string that holds the html code for an image, and setting a variable to hold that.
c. Add a header and a paragraph relevant to the image. Again, create two new variables, one for the header, and one for the paragraph.
d. In the function, use innerHTML for the div to change its content, by setting it to the 3 variables above, so that you change the image, the header, and the paragraph.
e. Now make the div clickable – add an onClick to the div tag that calls a function. Pass into the function's parameter the id of the div.
f. Add a prompt that asks, "would you like to see something else?"
g. If the user answers "yes" do the same thing over again – use the innerHTML of the div to change the picture, the header, and the paragraph.

Dynamic List:

Back to our list function, in which we are adding items to the list interactively, or dynamically. As it last stood, it looked like this:

```
function changeList2(par) {
        k = document.getElementById(par).innerHTML
        m = prompt("What item do you want to add to the list?")
        x="<li>" + m + "</li>"
        m = prompt("What item do you want to add to the list?")
        y="<li>" + m + "</li>"
        m = prompt("What item do you want to add to the list?")
        z="<li>" + m + "</li>"
        m= prompt("What item do you want to add to the list?")
        q="<li>" + m+ "</li>"
        v =  x + y + z + q
        document.getElementById(par).innerHTML = k+ v
}
```

What if you want to ask the user about whether the user wanted to add an item to the list at all? You can use another prompt and another if condition. This is what the code looks like:

```
v = ""
a = prompt("do you want to add an item to the list?")
if (a === "yes") {
        m = prompt("What item do you want to add to the list?")
        x="<li>" + m + "</li>"
        v = v + x
}
```

Step 1: (I will explain the code below – for now let's get it working) Copy your function, give it a new name, and add this to the new function. It should look something like this (don't include the line numbers):

```
1.  function changeList3(par) {
2.          k = document.getElementById(par).innerHTML
3.          v = ""
4.          a = prompt("do you want to add an item to the list?")
5.          if (a === "yes") {
6.                  m = prompt("What item do you want to add to the list?")
7.                  x="<li>" + m + "</li>"
8.                  v = v + x
9.          }
10.         m = prompt("What item do you want to add to the list?")
11.         y="<li>" + m + "</li>"
12.         m = prompt("What item do you want to add to the list?")
13.         z="<li>" + m + "</li>"
14.         m= prompt("What item do you want to add to the list?")
```

```
15.             q="<li>" + m+ "</li>"
16.             v =  v + y + z + q
17.             document.getElementById(par).innerHTML = k+ v
18.     }
```

Step 2: Make sure you modify 16. It will now add y, z, and q onto the end of v

Step 3: Save it and get it running. It should now ask the user whether they want to add an item to the list

Let's break this down:

3. **v = ""**

First, in this line (3), I created the variable v and set it to hold nothing. This is so we can join strings to it, depending on whether the user wants to add items to the list or not.

4. **a = prompt("do you want to add an item to the list?")**
5. **if (a === "yes") {**
6. **m = prompt("What item do you want to add to the list?")**

In lines 4, 5, 6 and 7 there's the prompt, and the variable a. The variable a will hold what the user types in in response to the prompt, "do you want to add an item to the list?" So it should hold "yes" or "no"

If the user answered "yes", then the code prompts the user for the item and put what the user typed in into the variable m.

7. **x="
" + m + "
"**

This line is the same as before – you're surrounding the m with <li> and tags.

8. **v = v + x**

And in line 8, v = v + x. It is saying, join the string in x onto the end of the string in v.

(Note: an even shorter way of saying it is v += x. Again, this is just laziness. It means the same thing – it means join x on to the end of the string v, or v now holds the empty string plus the list item joined together).

Step 4: Repeat this process for each of the different list items. The user should be prompted for whether they want to add another item 4 times, and each time, if they answer yes, they'll be prompted for what item they want to add and that item should be added to v. So, for example, my first two prompts looked like this:

```
k = document.getElementById(par).innerHTML
v = ""
a = prompt("do you want to add an item to the list?")
if (a === "yes") {
        m = prompt("What item do you want to add to the list?")
        x="<li>" + m + "</li>"
        v = v + x
}
a = prompt("do you want to add an item to the list?")
if (a === "yes") {
        m = prompt("What item do you want to add to the list?")
        x="<li>" + m + "</li>"
        v = v+x
}

…
```

NOTE that you can re-use the variables a and m because once they're used to generate the list item, you don't need them again, so it's okay to reuse them. You could also re-use x in this code because once x is strung onto the end of v, it's done with being useful, so you could put another value into it after that.

Step 5: Now you can remove line 16. As the code went along, the new list items were joined to the variable v, so you don't need to join them all together again at the end.

Step 6: Save and test. You should now be prompted 4 time for whether you want to add a list item, and if you respond, "yes" you should be prompted for what item you want to add.

Your Turn

1. (4) Make sure the code above works and you understand it.
2. (4) Choosing Old List: Include an if condition for the pre-existing elements on the page (the ones that get captured by the k variable, above. Use a prompt to ask the user, "Do you want to include the current list in your new list?" If the user says "yes", then include k in the list (in v) Otherwise don't include k. (Note that you will have to initialize v above the line that reads in k. Code occurs in order from the top down. You must initialize v before you can join it with anything).

That's it for Lab 3!! You're getting there! You can do it!!