Web Apps Assignment 6

The purpose of this assignment is to explore basic java script and interact with the DOM. Make sure that your commit messages are good. You can either make your commits and pushes from git bash or from GitHub desktop.

Part 1: Create remote and local repos

1. Click on the <https://classroom.github.com/a/GB12oPaI> and it will take you to GitHub classroom where you will have your own repo for the assignment. If this does not work, try copying the link and pasting it into your browser.
2. You should have a bare bones README.md and sample code for input and checked.
3. Clone down the repo to a local repo.
4. Open the repo in VS Code.
5. **Screen shot 1 of repo on VS Code for submission**

Part 2: Create basic HTML structure

1. Use VS Code to make a copy of sample-input.html with the name painter.html.
2. Pick a descriptive title and for the header use “Painter entry” followed by your name.
3. Save your work.
4. Open the file with a browser (like Chrome) and verify that you have the title on the tab and your header is displayed.
5. Inspect the code
6. **Screen shot 2 on your browser of rendered page and the inspector**.

Part 3: Create the input form and build the response code. This part is pretty close to the sample-input.html. You can modify it, or for more of a challenge, build up the elements on your own and refer back if you need help. Note that any error messages will show up in the console.

1. First, start with the form.
2. You will need two input elements each with an id
   1. The first element will hold a secret string value. Make it of type password. Give it an initial value of “Monet”. Label it with “The hidden painter name”
   2. The second element will hold the input to check. Make it of type text. Label it with “Guess the painter”.
3. Add the button. You can use either a regular button or an input of type button. It should have the text “Check the guess”. It will have an onclick that is checkPainter().
4. We need an element with an id to put our results. It can be a paragraph or a div.
5. We need some java script in a script element.
6. Create two global variables ***good*** and ***oops*** both initialized to 0.
7. Create a function checkPainter().
   1. It will get the value from the hidden painter input element.
   2. It will get the value from the guess input element.
   3. Compare the values and if they are the same add one to good and do an alert with message “You got it right”, otherwise add one to oops and alert with message “Sorry”
   4. Replace the innerHTML for the tagged element from step 4 with the good/oops counts.
8. Verify that the code works. You should be able to change both inputs and pressing the button should result in an alert and an updated score displayed.
9. Commit and Push. You can either use git bash or github desktop to do your commit and push operations.
10. **Screen shot 3 on VSCode with form and right/wrong holder visible.**

Part 4: Change an image

1. Get the three images for your plants from the last assignment and add them to your directory.
2. Add a div container and add one of the images to the container. Make the size of the image 100 by 100.
3. Add an onclick to the image with value selectImage().
4. Add a script. Make a global variable useImage with value 1
5. Create a function selectImage().
   1. Do a switch on the useImage with three cases (1, 2, 3)
   2. For each case set the image src and then update the useImage value to the next one. Wrap back around to the first image from the last one.
   3. Don’t forget the break
6. Test it out. Each click should cause the next image to display.
7. Commit and Push.
8. **Screen shot 4 on browser** (should see both the guess and image.)
9. **Screen shot 5 on VSCode** showing the java script.

Part 5: Get values and do a looping computation.

1. Use VS Code to make a copy of sample-input.html with the name conversions.html.
2. Pick a descriptive title and for the header use your name followed by “ shows currency conversions”.
3. You will need two input elements each with an id
   1. The base element holds the amount we are going to convert. Make it of type number. Give it an initial value of 50. Label it with “Enter base amount”
   2. The conversion element will hold the second value. Give it an initial value of 35. Make it of type number. Label it with “Enter the conversion rate.
4. Add the button. You can use either a regular button or an input of type button. It should have the text “Show conversions ”. It will have an onclick that is convert().
5. We need a tagged element to put our results. It can be a paragraph or a div.
6. We need some java script in a script element.
7. Create a function convert().
   1. It will get the base value from input amount.
   2. It will get the value from input conversion.
   3. It will initialize a local variable ***toShow*** to an empty string.
   4. Create a loop that will iterate 5 times. It will compute conversions based on five amounts: base, basex2, basex4, basex8 and basex16. Append each of the values and the converted value to toShow using a format of “***value*** is ***converted value***; “
   5. Replace the innerHTML for the tagged element from step 5 with “Conversions are:” concatenated with toShow
8. Check your results. An examples:
   1. **Conversions are: 10 is 350; 20 is 700; 40 is 1400; 80 is 2800; 160 is 5600;**  (base was 10, conversion is 35)
   2. **Conversions are: 3 is 0.9; 6 is 1.8; 12 is 3.6; 24 is 7.2; 48 is 14.4** (base was 3, conversion is 0.3)
   3. Commit and Push.

Part 6: Create a radio button group that switches between calculations.

1. Create a form holding four labeled radio input elements. You can use sample-checked.html as a reference. Each of the inputs will have onClick set to “chooseOp()”
   1. The first radio button will be labeled with “average”. Give it the value “mean”
   2. The second radio button will be labeled with “max”. Give it the value “maximum”
   3. The third radio button will be labeled with “min”. Give it the value “minimum”
2. We need a tagged element to put our results. It can be a paragraph or a div.
3. We need some java script in a script element.
4. Create a function chooseOp().
   1. It will get the value from input base from the previous part
   2. It will get the value from input conversion from the previous part
   3. It will set a value based on the checked property of the radio buttons and will use base and conversion as the operands to the corresponding operation. (I.E. if maximum is checked, then the value will be max of base and conversion).
   4. Replace the innerHTML for the tagged element from step 3 with appropriate text (E.g. max of 5 and 3.3 is 5)
5. **Screen shot 6 on browser** (should see both parts.)
6. **Screen shot 7 on VSCode** showing the java script.
7. Commit and Push.

Part 7: Use check boxes

1. Use VS Code to make a copy of sample-input.html with the name candy.html.
2. Pick a descriptive title and for the header use your name followed by “chooses candy”.
3. At the top of the body after the header add in a form group holding three labeled checkbox input elements. Each needs an id. (Note: you can keep the basic form, we will modify it in the next part.)
4. You will need three input elements each with an id each of checkbox.
   1. The first checkbox will be labeled with “Dark Chocolate”. Give it the value “True”
   2. The second checkbox will be labeled with “Nuts”. Give it the value “True”
   3. The third checkbox will be labeled with “Chewy”. Give it the value “False”
5. Add a button. You can use either a regular button or an input of type button. It should have the text “Select a box ”. It will have an onclick that is selectBox().
6. We need a tagged element to put our results. It can be a paragraph or a div.
7. We need some java script in a script element.
8. Declare a global variable my\_candy\_box and set it to an empty object.
9. Create a function selectBox().
   1. It will reset the my\_candy\_box to an empty object.
   2. It will set the properties dark, nuts, and chewy for my\_candy\_box with true or false depending on whether each of the checkboxs is checked or not.
   3. Replace the innerHTML for the tagged element from step 6 with my\_candy\_box.
10. Commit and Push.

Part 8: Work with the object from the previous part

1. Below the checkbox form, we will need a form (you can modify the form from the starter code) the with two text elements and a button.
2. You will need two input elements each with an id
   1. The first element holds the name of a property. Make it of type text. Give it an initial value of “size”. Label it with “Enter the size”
   2. The second element will hold the price of the box. Make it of type text. Label it with “Enter price”.
3. Add the button. You can use either a regular button or an input of type button. It should have the text “Update the box”. It will have an onclick that is updateBox().
4. Add in a function updateBox() to the script from the previous part.
   1. It will get the value from size input element.
   2. It will get the value from the price input element.
   3. It will set the properties of my\_candy\_box with the value we just got from the input elements. (Use the form my\_candy\_box[…] = …)
   4. Replace the innerHTML for the tagged element with my\_candy\_box.
5. Commit and Push.
6. **Screen shot 8 on browser**.
7. **Screen shot 9 on VSCode** showing the java script.

Part 9: Host on GitHub pages

1. Go to setting on GitHub and select the main branch.
2. Copy the URL.
3. Edit README and add three lines “Hosted at “ followed by a link to the URL from the previous step with the files painter.html, conversion.html, and candy.html. (Commit and Push if you did it on the local repo.)
4. Verify that if you click on the links in the README, that you get your pages.

**Bonus**: On the first part, if the guess is incorrect for 10 times, replace the onclick property value with a new function call that displays an alert with the text “Sorry, ran out of guesses.” You may also pull in the bootstrap framework and use it to style the elements in these apps. Use your sense of style to improve the look of these apps. Do small chunks and commit each time your modification succeeds. I would recommend that you tag the commit for your working app. (Read about creating a Git tag for a commit here <https://git-scm.com/book/en/v2/Git-Basics-Tagging>)

**Provide screen shots on the browser showing each of the modified apps.**