# Process Notes

I downloaded the ZIP from GitHub, then reviewed the files and opened the Resources folder. I copied the README file into a blank doc to use as a checklist and a place to record my process as I work. I used Atom text editor to open the .html file and begin working through the task list in order, writing these notes as I go and checking my work periodically by loading the index.html file in Google Chrome. I had trouble with getting the header to be fixed but not overlap any other elements (various workarounds I tried are documented here). After completing the tasks, I committed the files to a new Github repository.

HTML Tasks

CSS Tasks

Javascript Tasks

## HTML Tasks

**Reference the jellyfish file**

The jellyfish image is already referenced in the banner, but as a .png file. It is a .jpg file, so I change the reference to reflect that.

1. Jellyfish file is a .jpg image: enter <img> tag
2. Add the src attribute
3. Reference the path to the file: <img src= “Resources/jellyfish.jpg”>

**Reference the main.js file**

1. In the header, enter script tags.
2. Point to the main.js file by adding the src attribute to the opening tag and specifying the file name: <script src= “main.js”></script>.

**Reference the styles.css file**

1. In the header, enter <link> tag.
2. Specify the type of file with the rel attribute.
3. Add href= “” to link to the file.
4. In the href attribute, specify the file path for styles.css.
   <link rel=”stylesheet” href="styles.css">

**Change the href to link to Veeva.com**

1. Use Ctrl+F to find the href.
2. Paste the veeva.com URL between the quotation marks in the href to make the link active.

**Modify the Click Here button to open Veeva.com as a new tab**

1. In the <a> tag after the href attribute, add the target attribute.
2. Set the target attribute to = “_blank” to open the link as a new tab.
3. Save file and check work so far by opening the html file in Google Chrome and testing links.

**Separate the lorem ipsum content into two columns**

I made the text into two columns by styling them to each take 50% of the page, and then floating each column to the correct side of the page. I knew I could add the styling for each of these classes either internal, within <style> tags at the top of the document, or external, in the styles.css file. I chose external because it makes for slightly faster load time and reduces file size. It also makes the code tidier/easier to read for me.

1. Separate the text at line 24 (4 paragraph tags in each column).
2. Add a closing tag for the first div.

3. Add a new div tag after the closing tag.
4. In the original div tag, add class= “left”.
5. In the new div tag, add class= “right”.
6. Open styles.css.
7. At the end of the style sheet, add classes for .left and .right.
8. Use the width attribute to set the column width to 50% (half the page) for each div.
9. Use the float attribute to place the left column on the left side of the page, and the right column on the right side of the page.
10. Refresh the html file in Chrome to check work.

**Build a form**

1. Below the header div, enter the <form> tags.
2. Below the opening form tag, add a header for the form at h2 level (“Here’s the form!”)

**Add a checkbox**

1. Enter the <input> tag, then set attribute type to “checkbox”.
2. To link the checkbox with its label, add the id attribute in the input tag. Set the id to “box1.”
3. So that a server could process and read the information in this form, add a name attribute in the input tag (box1), and then added a value attribute and set that to “1.”
4. After the input tag, add the label tag and link it the input element by adding attribute “for” and setting it to “box1.” In between the opening and closing tags for the label, add the text that should appear next to the box (“Subscribe to newsletter”)
5. Enclose the checkbox section in a div to keep it separate from the next sections in the form.

**Add a picklist**

1. Create a new div below the checkbox div.
2. Enter the <select></select> tags to create a picklist, and set the name attribute to “picklist.”
3. Between the select tags, enter the <option></option> tags to create the first option.
4. Add the attribute “value” in the first option tag and set the value to “inventory monitoring.”
5. Enter “Inventory Monitoring” between the two option tags to have this text show as the label.
6. Save the file and refresh the page in Chrome to check my work.
7. Since the code works, I copy/paste the option code twice, to create my two other options for the picklist.
8. Modify the value and label text to on each of the copies to reflect the remaining two options, order management and MyInsight.
9. At the top of the div, add the text “I’m interested in…” in paragraph tags to give some context for the picklist.

**Add a text field**

1. Create a new div.
2. Add the input element tags and set the type attribute to text. Set a unique id (“color”), and create a unique name (“favorite color”).
3. After the closing input tag, add the label tag. Link the label to the input element using the for attribute, then enter the text “My favorite color is:” between label tags. This text will be displayed next to the text field.
4. Refresh page to check work.

**Add a submit button**

1. Create a new div.
2. Add input element and set type attribute to “submit”.
3. Refresh page to check work.

# CSS Tasks

**Fix the Header Bar so it "sticks" to the top of the page when you scroll (it needs to be static)**

1. Find .header-bar class in styles.css.
2. Set position to "sticky" and set top:0 so that it will stay at the top of the page.
3. Check work in Chrome.

When I check my work, the header doesn't stay in the same place over the body text, so I go back to the css to try a different position attribute.

4. Return to styles.css and the header-bar class.
5. Change the position to "fixed".
6. Set z-index to 1 to bring the header to the front, over the body text.

**Modify the Click Here button to change color when your cursor hovers over it**

1. Add a pseudoclass for a.btn:hover so that the color of the button only changes on hover.
2. Add attribute color, and set to blue.
3. Check work in Chrome; this styling isn't working.

I go back and read through the rest of the css for the button, and realize that the button's color is set using the background-color attribute, not color (color only changes text color).

4. In the a.btn:hover styling, delete color attribute and add background-color: blue.
5. Check work.

**Fix the Header Bar to never overlap over the other elements**

1. Add a div (class="container") to hold all of the other elements below the header bar.
2. In the CSS, position the container relative and set the padding-top to 144px (header height).

I expected this to set the content apart from the header and allow the header to stick at the top while the content shifted below, but instead the header bar continued to overlap the elements.

3. Try to adjust the margin-top to fit header height (didn't work).
4. Undo and try positioning the container div absolute, setting top to 144px and bottom to 0 (didn't work).
5. Undo and try to change overflow to scroll, hoping that changes the spacing (didn't work).
6. Undo and try adding padding-top to all other divs and position relative (didn't work).
7. Undo and try adding float:top to the header-bar.
8. Undo and change the header bar's z-index to -99 to go behind all the other elements.

I'm guessing this isn't the correct answer, but it does accomplish the task of making a fixed header not overlap content, at a basic level?

**Modify the hero image to span the full width of the page**

1. In the html, find the div it's in: class=banner.
2. Find .banner in styles.css.
3. Below .banner, add attribute width and set it to 100%.

**Change the Click Here button from blue to Veeva Orange**

1. Use the developer tools feature on Google Chrome to look at the code on Veeva.com homepage and find the hex no. for Veeva orange (#F8991D).
2. In styles.css, change the background-color on the .button class to #F8991D.
3. Change color: to white from transparent, so the text shows up on the orange background.

**Display the This is normally hidden! text on the header bar**

1. Look through the text tags in html to find what element the text is in (.header-bar p).
2. Find corresponding .header-bar p class in css.
3. Change display attribute from display: none to display: inline.

**Left align the This is a Test Site text on the top**

In .header-bar class, change text-align from center to left.

**Center the Hello! text within the hero image**

1. Identify class from html: .content.
2. Add text-align:center to .content class in styles.css file.

**Set a *light gray border* around the two columns**

1. In the css class for the left div, add border-style:solid to create a border.
2. Add border-color:lightgrey to set the border color.
3. Copy/past border style and color attributes into .right class to apply to right column.
4. Adjust width for both columns to 49% to allow room for borders.

**Change the *font* of the text to Roboto**

1. Change font-family from Arial to 'Roboto'.
2. Add ", sans-serif" specify broader font family.
3. In the head of the html, embed the Google fonts link to Roboto font to ensure Roboto shows up on most computers.

**Make the right column *link* to Veeva.com**

1. Before the right column div, enter an opening <a> href tag with a link to veeva.com.
2. Enter the closing </a> tag at the end of the div. The whole column now acts as a link.

**Change the *background* of the right column from white to gray when hovering**

1. Create pseudoclass .right:hover.
2. Add attribute background-color and set it to :grey.

***Center* the form**

1. In the css for the form element, set margin attribute to : 0 auto. This centers the element by giving margins of 0 on top and bottom and equal margins on the sides.
2. Set text-align attribute to center, to center the text as well.

**Set the *color* of the form to gray**

Under the form element in styles.css, add the attribute background-color and set it to grey.

**Give the form a *border radius***

Under the form element in styles.css, add the attribute border-radius and set it to 40 px (for visibly rounded edges).

**Ensure that each *input* is on a separate line**

I check the form in my browser: I already used  and <br> to put each input on a separate line.

**Set the submit button to take up the entire width of the form**

1. Add the input element to the styles.css sheet.
2. Add [type=submit] after input, to apply the subsequent styling to only the submit button.
3. Under input[type=submit], add width attribute and set it to 100%.

# Javascript Tasks

### Link the JQuery library

1. Add <script> tags in the head of the document.
2. Copy the most recent jQuery library link from Google Hosted Libraries.
3. Paste the jQuery link in the opening script tag, prefaced by the src attribute and enclosed in quotes.

I could also download jQuery as a .js file to my computer and then reference it in the head, like how I referenced the main.js file. However, using this link to the library saves space on my computer and also may make the page load faster for users, who probably already have the jQuery library in their cache.

### Output some text in an alert whenever the site loads

1. Add <script> tags in the head of the document.
2. After the opening script tag, add “function,” then give the function a name ending in “Alert()” to use Java’s native alert function (I chose loadAlert).
3. On the next line, specify the text to show in the alert by adding “alert,” followed by the text to show in parentheses and quotes.
4. Make the script execute when the page loads by specifying the event “onload” in the first <body> tag, and setting it to ”showAlert()” on load. This way, the alert function that I put in the head will show up when the body has loaded.

### Create an alert that displays the values of the form when selecting the submit button

1. Add <script> tags in the head of the document.
2. In the new script tags, add “function,” then give the function a name ending in “Alert()” to use Java’s native alert function (I chose submitAlert for this one).
3. In the <input tag for the submit button, add an “onclick” event to make the alert load when the submit button is selected. Connect the onclick event to the function by specifying “onclick= “submitAlert()”.
4. Set up the variables to pull the values from each of the three elements in the form: Start by specifying var = for the id for the first input element to pull from. Then add the statement document.GetElementByID and set it to the id of the element to pull from (“box1”), then pull the values values from that element by adding .value at the end of the statement.
5. Copy/paste and fill in relevant ids for the remaining two elements.
6. After the variables, add alert () to create a space for text in the alert.
7. Set the alert to show the output of the three variables just written by putting the variable names, joined by “+”, in the parentheses following “alert”.
8. Add labels for the values in the alert by entering the label in quotations before each variable name.
9. To set each result on a different line, add line breaks (\n) at the beginning of the last two labels, within the quotation marks.